MASTER EN SCIENCES MATHÉMATIQUES

Approche spectrale pour la classification d'états de conscience

Delaunois, Bastien

Award date:
2018

Awarding institution:
Universite de Namur

UNIVERSITE DE NAMUR

Faculté des Sciences

APPROCHE SPECTRALE POUR LA CLASSIFICATION D'ETATS DE CONSCIENCE

Mémoire présenté pour l'obtention

du grade académique de master en « Sciences Mathématiques à finalité spécialisée en perspectives professionnelles des mathématiques appliquées »

Bastien DELAUNOIS
Promoteur : Alexandre MAUROY

Juin 2018

Remerciements

Le fruit du travail est le plus doux des plaisirs. - De Vauvenargues

J'aimerais commencer ce mémoire en remerciant mon promoteur, Alexandre Mauroy, pour sa disponibilité, sa patience, sa gentillesse et ses encouragements. Merci à lui qui, encore plus ces derniers jours, a pris de son temps pour lire et relire ce manuscrit et répondre à mes questions, parfois triviales.

Je voudrais également remercier Johan Barthelemy, Timoteo Carletti et André Hardy de faire partie de mon jury, et de prendre le temps de lire ce travail.

Je remercie également Raphaël Liegeois, qui a pris la peine de répondre à certaines de mes questions concernant les données traitées dans ce mémoire.

La famille c'est une richesse incroyable, ça donne les des outils pour pouvoir affronter les moments extraordinaires, les moments les plus difficiles, les hauts, les bas. - Céline Dion

J'aimerais également remercier ma maman, ma sœur et mes grands parents pour leur présence, leurs encouragements, leur soutien et leur confiance. Je m'excuse de tout le stress que j'ai pu provoquer au fil des ans, lors de l'attente des résultats. Un énorme merci supplémentaire à ma maman qui, malgré que « chaque fois tu me donnes tes travaux à relire la veille », malgré que ce mémoire « est plus gros que ton travail de Français de rhéto », a relu ce manuscrit.

Les amis sont des compagnons de voyage qui nous aident à avancer sur le chemin d'une vie plus heureuse. - Pythagore

C'est vous, mes amis, que je remercie maintenant. Mes amis qui, sans vous, je ne sais pas ce que je serais, vous qui me faites rire, sourire, pleurer de joie. Vous qui m'avez motivé dans les moments où j'avais envie de baisser les bras. Vous qui m'avez encouragé et donné une volonté et une raison supplémentaires de vaincre les secondes sessions. J'aimerais plus particulièrement vous remercier, compagnon de guindaille, compagnon de blocus, compagnons depuis le début, toi, *le frère que j'ai choisi*, et toi, *qui sais le démontrer.*

Enfin, je remercie toute personne qui, de près ou de loin, m'a permis d'en arriver là où j'en suis.

Merci.

Résumé

Détecter l'état de conscience d'une personne dans un état végétatif à partir de données d'imagerie médicale (par exemple l'imagerie par résonance magnétique fonctionnelle - IRMf) est un grand enjeu. Il existe actuellement des techniques telles que celles basées sur les corrélations entre les régions du *Default Mode Network*, mais leurs résultats restent cependant très limités. Nous présentons ici une nouvelle méthode qui, contrairement aux méthodes précédentes, exploite la dynamique du réseau sous-jacent aux mesures IRMf. Cette approche est basée sur la théorie de l'opérateur de Koopman et sur le lien existant entre le spectre de l'opérateur de Koopman et le spectre du réseau. L'hypothèse soutenant cette méthode est que, lorsque un patient change d'état de conscience, le réseau subit également un changement, ce qui implique une variation du spectre mesuré.

Dans ce travail, nous avons combiné la méthode de décomposition en modes dynamiques, permettant d'extraire le spectre de l'opérateur, à une méthode de classification *Extra-Trees*. En proposant des choix de variables pertinents, nous avons proposé différentes méthodes que nous avons comparées. Les résultats obtenus sont encourageants et démontrent, de manière préliminaire, que notre approche spectrale est prometteuse pour la détermination des états de conscience.

Mots-clés : Opérateur de Koopman, IRMf, *Default Mode Network*, Classification, *Extra-Trees*, décomposition en modes dynamiques, état de conscience, *Dynamic Causal Modelling*.

Abstract

A crucial challenge in the field of neuroimaging (e.g. functional magnetic resonance imaging - fMRI) is to detect the state of consciousness of patients in vegetative state. Current methods use for instance correlation between *Default Mode Network* regions. However, these methods yields limited results. The method we develop in this master's thesis exploits the dynamics of the underlying network given by fMRI data. This approach is based on the Koopman theory and the connection between the Koopman operator spectrum and the network spectrum. The main assumption of this method is that changes in patient's consciousness level implies changes in the network and therefore in the measured spectrum.

In this work, we combined the dynamic mode decomposition method, which extract the operator spectrum, with the Extra-Trees classification method. Using a relevant set of variables, we proposed several methods and compared them. The obtained results are promising and support our hypothesis that spectral dynamical properties are relevant in the context of consciousness level classification.

Keywords : Koopman operator, fMRI, Default Mode Network, classification, Extra-Trees, dynamic mode decomposition, consciousness level, Dynamic Causal Modelling

Table des matières

Introduction

La capacité d'un médecin à pouvoir déterminer l'état de conscience d'un patient dans un état végétatif est cruciale. En effet, cette information est capitale dans les prises de décisions vis-à-vis du patient, qui peut par exemple être soumis à un traitement adapté. Prenons l'exemple d'un patient se trouvant dans un état végétatif, souffrant du *locked-in syndrom*, aussi connu sous le nom de syndrome d'enfermement. Dans le cas classique du syndrome, ce patient est atteint d'une immobilité complète, à l'exception des mouvements verticaux du regard et des paupières [7]. Le diagnostic d'un tel état n'est pas toujours évident à faire. Cependant, au plus tôt il est réalisé, au plus vite les soins rééducatifs peuvent commencer, ce qui permet de réduire le taux de mortalité de tels patients [18]. Cet exemple illustre bien l'importance d'avoir des méthodes efficaces et fiables de détermination de l'état de conscience.

De nos jours, il existe certaines méthodes de détermination de l'état de conscience d'un patient. Certaines d'entre-elles reposent sur des critères physiologiques, comme par exemple l'échelle de Glasgow qui se base sur l'ouverture des yeux, la réponse verbale et la réponse motrice pour attribuer un stade de coma à un patient, ou une de ses variantes, l'échelle de Glasgow Liège, qui ajoute les réflexes du tronc cérébral aux précédents critères.

Une autre méthode utilisée afin de déterminer les différents états de conscience d'un patient se base sur le *Resting State Network*. En effet, le cerveau, même lorsqu'il n'est pas sollicité, est toujours en activité. Plus particulièrement, certaines de ses zones sont toujours actives, notamment pour les fonctions essentielles à notre survie. Ces régions sont interconnectées grâce à la matière blanche, et sont regroupées sous l'appellation *Resting State Network*. Ce réseau est en réalité composé de plusieurs sous-réseaux. L'un d'entre-eux est le *Default Mode Network* (DMN), dans lequel nous pouvons observer des changements dans la connectivité selon l'état de conscience dans lequel se trouve le patient. Cette méthode essaye donc de déterminer l'état de conscience du patient grâce aux corrélations qui existent entre les signaux d'imagerie par résonance magnétique fonctionnelle mesurant l'activité des différentes régions du DMN [16]. Cette approche a l'avantage d'une interprétation facile. En effet, elle nous donne des réponses directes quant aux hypothèses que nous pourrions faire sur la connectivité fonctionnelle d'une ou de plusieurs régions considérées. Le principal inconvénient de cette méthode statistique est qu'elle considère le signal dans chaque région comme une fonction « statique ». Elle ne prend donc pas en compte la dynamique.

Notre approche est différente de celle utilisant les corrélations et consiste à développer le *Proof of Concept* d'une méthode exploitant la dynamique du cerveau, via la théorie de l'opérateur de Koopman. Nous essayerons de développer des outils permettant de classer ces états de conscience sur base de propriétés du réseau. Exploitant le fait qu'il existe un lien entre le spectre

de l'opérateur de Koopman et le spectre du réseau [22], nous appliquerons l'algorithme *Dynamic Mode Decomposition* (DMD) sur les données issues de l'IRMf. En effet, cet algorithme permet d'obtenir le spectre de l'opérateur de Koopman. Nous pourrons donc comparer ce spectre et appliquer ensuite différentes méthodes de classification sur celui-ci, afin de pouvoir déterminer l'état de conscience des patients soumis à l'IRMf.

Les données dont nous disposons ont été récoltées via IRMf par des médecins de l'équipe du Docteur Steven Laureys du groupe *Coma Science Group* de l'Université de Liège. Pour obtenir ces données, les médecins ont administré un sédatif aux patients et ont procédé à des relevés de mesures lorsque ceux-ci étaient dans quatre états de conscience différents : (1) conscient, c'est-à-dire avant l'administration, (2) moyennement endormi, (3) en sommeil profond, (4) en phase de réveil. Ces données concernent le signal BOLD, *Blood Oxygen Level Dependant*, de 28 régions et ont été récoltées toutes les $2,46$ secondes. Ces régions ont été obtenues par agrégation des voxels (pixels 3D) mesurés par l'IRMf et qui ont un comportement semblable. Les données dont nous disposons ont été fournies sous trois formes, chacune d'entre-elles correspondant à une phase de pré-traitement du signal BOLD. L'objectif de ce mémoire est d'être capable de détecter les quatre états de conscience représentés dans ces données.

Notons que dans ce travail, nous parlerons de coma ou d'état végétatif comme étant un seul et même état de conscience, assimilé dans le cas de nos données au sommeil profond.

Ce mémoire se divise en quatre chapitres. Les deux premiers présentent les outils dont nous avons besoin dans les deux derniers chapitres. Dans le premier chapitre, nous introduisons la technique d'imagerie par résonance magnétique fonctionnelle (IRMf). Nous y présentons également un modèle, *Dynamic Causal Modelling*, qui permet une simulation du signal BOLD obtenu via IRMf. Le second chapitre présente les concepts de la théorie de Koopman, ainsi que l'algorithme de décomposition en modes dynamiques. Nous développons également le lien entre le spectre de l'opérateur de Koopman et celui d'un réseau. Finalement, nous mentionnons différentes techniques de classification, mais essentiellement celle des *Extra-Trees*, que nous utiliserons principalement. Dans le troisième chapitre, nous réalisons des simulations de données IRMf. Nous aurons ainsi des données pour dix patients pour deux états de conscience. Ensuite, nous entrainerons notre modèle grâce aux outils développés aux chapitres précédents. Dans un premier temps, nous appliquerons l'algorithme de décomposition en modes dynamiques sur ces simulations, afin d'obtenir le spectre de l'opérateur de Koopman. Nous sélectionnerons ensuite des variables qui serviront à la classification de ces états. Dans le dernier chapitre, nous commencerons par présenter plus en détail les données réelles issues de l'IRMf. Nous appliquerons ensuite le même procédé qu'au troisième chapitre. En effet, nous commencerons par extraire le spectre de l'opérateur de Koopman, grâce à l'algorithme DMD, et nous sélectionnerons des variables afin d'entrainer notre modèle, en appliquant dans un premier temps celles déterminées par l'entrainement sur les simulations. Nous essayerons ensuite de caractériser les performances de la méthode développée, c'est-à-dire déterminer si elle permet la détection des états de conscience ou si elle est plutôt précise. Nous terminerons ce chapitre par une brève comparaison entre les résultats obtenus grâce à notre modèle basé sur les *Extra-Trees*, et ceux basés sur un arbre de décision ou une régression logistique.

Chapitre 1

Imagerie par résonance magnétique

Dans ce chapitre, nous décrivons la méthode d'imagerie par résonance magnétique fonctionnelle, qui nous permet d'obtenir les données hémodynamiques des patients. Nous les considérerons dans la deuxième partie de ce travail. Ensuite, nous présentons la modélisation *Dynamic Causal Modelling*, qui fournit un modèle permettant la simulation des données semblables à celles obtenues via l'imagerie par résonance magnétique fonctionnelle (IRMf).

1.1 Méthode d'imagerie par résonance magnétique fonctionnelle

Le principe de l'imagerie par résonance magnétique classique est de fournir des images du corps humain grâce aux propriétés magnétiques du noyau d'hydrogène présent dans l'eau, qui est l'une des principales composantes des tissus biologiques. Pour ce faire, un patient est soumis à un champ magnétique qui contraint les noyaux d'hydrogène à s'aligner dans la direction de ce champ. Cela crée des signaux magnétiques qui s'assemblent afin de former un signal plus grand et mesurable.

L'imagerie par résonance magnétique fonctionnelle, IRMf, est une technique d'imagerie utilisée, entre autres, pour étudier le fonctionnement du cerveau. Si dans un premier temps elle nécessitait l'injection d'un traceur radioactif, le gadolinium, la méthode a évolué au fil des années ; elle se base désormais sur l'observation de la variation de volume, de débit et d'oxygénation du sang, c'est-à-dire les variations hémodynamiques, dans les régions considérées [17]. Elle ne nécessite donc plus aucune injection. Dans le cadre d'une application au cerveau, l'IRMf exploite la diminution en désoxyhémoglobine, l'hémoglobine déchargée en oxygène, en aval, après le passage du sang dans les neurones activés. En effet, lors de leur activation, les régions cérébrales sont sujettes à une arrivée de sang oxygéné, comme montré sur la FIGURE 1.1. Cet apport d'oxygène est supérieur aux besoins réels de la région cérébrale activée, c'est pourquoi la consommation d'oxygène est proportionnellement inférieure à son apport. Ainsi, le rapport local entre l'oxyhémoglobine et la désoxyhémoglobine va être modifié et un signal magnétique mesurable va apparaître, c'est le signal BOLD. L'hémoglobine possède des propriétés magnétiques différentes selon qu'elle soit oxygénée ou non. En effet, lorsque l'hémoglobine se débarasse de l'oxygène au niveau des capillaires cérébraux, une diminution de la quantité de fer a lieu, et celui-ci se retrouve à l'état d'ion (Fe^{2+}), ce qui laisse deux électrons libres dans la molécule de désoxyhémoglobine [1]. Ce sont ces électrons qui sont à l'origine du magnétisme de la désoxyhémoglobine et qui génèrent une modification de champ magnétique local.

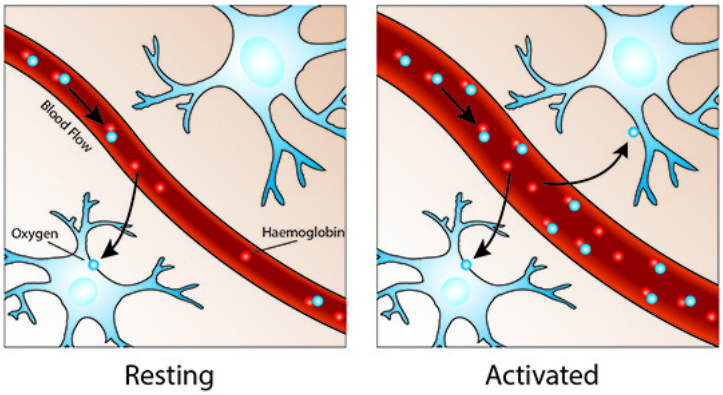


FIGURE 1.1 – Illustration de la différence entre l'apport d'oxygène dans une région cérébrale activée (à droite) et non-activée (à gauche). Nous pouvons observer l'afflux sanguin dans la région activée. Source : [26].

Cette méthode d'imagerie possède des avantages indéniables. Le premier est qu'elle n'est pas invasive, c'est-à-dire aucune lésion de l'organisme n'est nécessaire. Ensuite, elle ne nécessite aucun rayonnement, ce qui la rend saine pour le patient. Elle a une bonne précision spatiale et, bien que moins précise que l'électroencéphalogramme, sa précision temporelle est également très bonne. Enfin, c'est une technique qui est facile à utiliser. Concernant la limitation de l'imagerie par résonance magnétique fonctionnelle, elle n'est pas technologique mais physiologique. En effet, les éléments du cerveau prennent plus de temps à réagir à une activation que l'IRMf à capturer ces changements. Notons que cette méthode a été utilisée dans beaucoup d'études qui visent à mieux comprendre le fonctionnement d'un cerveau sain, ainsi que dans un nombre de plus en plus élevé d'études visant à comprendre la manière dont certaines fonctions cérébrales sont interrompues en cas de maladie.

Bien que que l'IRMf soit une forme récente d'imagerie cérébrale, l'idée sous-jacente à cette technique remonte à la fin du XIX$^{\text{ème}}$ siècle. Comme le présente le site internet d'Oxford [26], le scientifique italien Angelo Mosso avait déjà, à l'époque, pensé à déduire l'activité cérébrale en mesurant les variations de flux sanguin. Mais, ce n'est qu'à la fin des années nonante que la méthode commencera à se développer.

1.2 Dynamic causal modelling

Le but de la modélisation *Dynamic Causal Modelling*, notée DCM, est de modéliser les interactions entre les différents neurones, à l'aide de séries temporelles issues de l'imagerie cérébrale. Le but de cette modélisation est d'estimer et déduire le couplage entre les régions du cerveau et son avantage par rapport aux autres approches de modélisation est qu'elle adopte la nature dynamique et non-linéaire des réponses cérébrales, ce qui permet d'être au plus proche de la réalité. Cependant, nous ne voulons pas déduire de couplage, mais obtenir une dynamique pour un réseau considéré. Nous utiliserons donc la modélisation DCM dans le sens inverse de ce pour quoi elle a été créée, c'est-à-dire que nous imposerons le couplage entre les différentes régions, pour pouvoir en déduire la dynamique.

Le modèle *Dynamic Causal Modelling* décrit dans [13] est un modèle *entrée-état-sortie* avec plusieurs entrées et plusieurs sorties. Les entrées correspondent à des fonctions stimulus, les états couvrent les activités neuronales, et les sorties correspondent aux réponses hémodynamiques des régions cérébrales considérées. Le modèle considère donc le cerveau comme un système dynamique non-linéaire soumis à des entrées et produisant des sorties.

Nous savons que toute activité neuronale, dans toute région du cerveau, provoque un changement de volume des vaisseaux sanguins, ainsi qu'un changement de quantité en désoxyhémoglobine. Ces changements, comme expliqué dans la section précédente, permettent d'obtenir le signal BOLD, *Blood Oxygen Level Dependant*, grâce à l'IRMf. C'est ce signal qui sera la sortie du modèle DCM.

Le but de cette section n'étant pas de montrer toute la construction du modèle DCM mais de présenter brièvement ses constituants, nous n'entrerons pas dans tous les détails. Notons que nous présenterons DCM pour une utilisation par rapport à l'imagerie par résonance magnétique fonctionnelle, comme représenté à la FIGURE 1.2.

La première composante de DCM est l'équation d'état neuronal, dont les détails de construction peuvent être consultés dans l'article [27]. Considérons un système de n régions cérébrales inter-connectées. Nous pouvons modéliser les changements temporels d'un vecteur d'état neuronal

$$x(t) = \begin{bmatrix} x_1(t) \\ \cdots \\ x_n(t) \end{bmatrix},$$

où chaque région du système est représentée par une seule variable d'état $x_i(t)$, à l'aide de la forme bilinéaire de l'équation d'état générale pour les systèmes déterministes non-autonomes, c'est-à-dire pour les systèmes qui échangent de la matière, de l'énergie ou de l'information avec leur environnement, présentée dans [27]. Cette forme bilinéaire s'écrit

$$\begin{aligned} \frac{dx}{dt} &= F(x, u, \theta^{(n)}) \\ &= \left(A + \sum_{j=1}^{m} u_j B^{(j)} \right) x + Cu, \end{aligned} \tag{1.1}$$

où $u(t)$ est une fonction d'entrée à m composantes, qui correspond aux fonctions stimulus. Les paramètres neuronaux

$$\theta^{(n)} = \left\{ A, B^{(j)}, C \right\}$$

peuvent être exprimés comme des dérivées partielles de la fonction F, c'est-à-dire

$$A = \left. \frac{\partial F}{\partial x} \right|_{u=0}, \quad B^{(j)} = \frac{\partial^2 F}{\partial x \partial u_j}, \quad C = \left. \frac{\partial F}{\partial u} \right|_{x=0}.$$

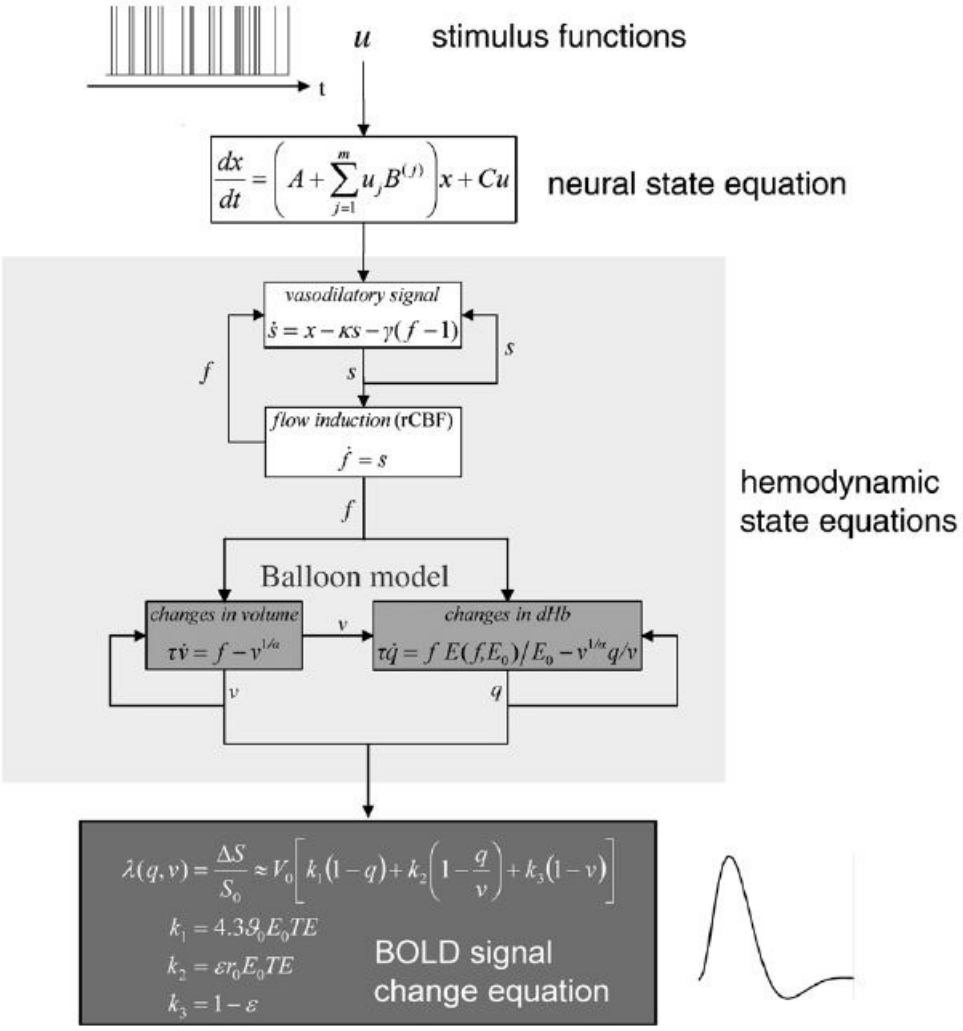


FIGURE 1.2 – Représentation schématique de la modélisation DCM. Les fonctions d'entrée u provoquent des réponses neuronales x. Ces réponses déclenchent une série d'équations hémodynamiques, et les réponses venant de ces équations permettent d'obtenir le signal BOLD.
Source : [27].

La matrice A, matrice Jacobienne ou matrice de connectivité, représente la connectivité entre les régions, en l'absence d'entrée. Autrement dit, la matrice A représente la matrice d'adjacence du réseau, c'est-à-dire que l'élément A_{ij} de cette matrice correspond au nombre d'arêtes entre le nœud i et le nœud j. Les matrices $B^{(j)}$ correspondent aux changements de connectivité induits par la $j^{\text{ème}}$ entrée u_j. La matrice C, quant à elle, intègre les influences directes des entrées sur l'activité neuronale.

Ces paramètres neuronaux $\theta^{(n)} = \left\{A, B^{(j)}, C\right\}$ définissent donc les interactions entre les régions du cerveau, à un niveau neuronal. Notons que c'est principalement la matrice A qui nous intéressera, puisque nous essayerons de détecter les changements dans les couplages.

Maintenant que nous avons présenté la première partie de la modélisation, nous allons présenter les équations hémodynamiques, comme le montre la FIGURE 1.2. En effet, afin de traduire l'activité neuronale en signal, nous devons coupler le modèle de dynamique neuronal au modèle hémodynamique, comme expliqué dans [27].

La deuxième partie de la modélisation DCM s'occupe de la dépendance aux changements de flux sanguin et de la quantité de désoxyhémoglobine induite par rCBF, le flux sanguin des régions cérébrales, du signal BOLD. Tout comme pour les équations d'état neuronal, nous ne présenterons pas tous les détails de cette deuxième partie. Ceux-ci, ainsi que certaines variantes, sont consultatbles dans les articles [27] et [15].

Les réponses vasculaires de l'activité neuronale correspondent à un oscillateur amorti. En effet, les changements dans l'activité neuronale x provoquent une diminution exponentielle du signal vasodilatateur s, qui est lui-même soumis à une régulation du flux $f_{entrant}$, qui est représentée par les équations d'état neurovasculaire

$$\begin{aligned} \frac{ds}{dt} &= x - \kappa s - \gamma(f_{entrant} - 1), \\ \frac{df}{dt} &= s, \end{aligned}$$

où κ est la constante de vitesse à laquelle décroît le signal s et γ la régulation due au flux $f_{entrant}$. Notons que ce modèle est linéaire et modélise la relation entre l'activité neuronale et rCBF, et que l'activité neuronale x dont il est question dans ce modèle correspond à la sortie des équations neuronales.

Comme représenté sur la FIGURE 1.2, c'est après les équations vasculaires qu'intervient le modèle Balloon. Celui-ci nécessite la connaissance de l'évolution du volume sanguin ν ainsi que de la quantité en désoxyhémoglobine q. Il fournit des équations différentielles pour ν et q en se basant sur deux hypothèses. La première est que le changement de volume sanguin ν dans les veines est la différence entre les flux sanguins entrant et sortant, moyennant une constante temporelle, c'est-à-dire

$$\tau \frac{d\nu}{dt} = f_{entrant}(t) - f_{sortant}(\nu),$$

où τ est le temps moyen que prend le sang pour traverser le compartiment veineux (c'est-à-dire la quantité de sang présente dans les veines) et correspond au quotient du volume sanguin résiduel par le flux sanguin résiduel, c'est-à-dire

$$\tau = \frac{V_0}{F_0}.$$

Nous pouvons visualiser cette première hypothèse par le fait que les petits vaisseaux sanguins réagissent à une augmentation du flux sanguin entrant de la même manière qu'un ballon que nous gonflons, d'où le nom de ce modèle.

La deuxième hypothèse permet d'obtenir l'équation de la quantité en désoxyhémoglobine q. En effet, elle stipule que l'extraction de l'oxygène est liée au flux sanguin. Le changement de cette quantité q correspond à l'apport en désoxyhémoglobine dans les compartiments veineux, duquel est soustrait la quantité expulsée. En supposant avoir de l'oxyhémoglobine dans les vaisseaux précapillaires, nous avons

$$\tau \frac{dq}{dt} = \underbrace{f_{entrant}(t)\frac{E(f_{entrant}, E_0)}{E_0}}_{\text{Apport en désoxyhémoglobine}} - \underbrace{f_{sortant}(\nu)\frac{q(t)}{\nu(t)}}_{\text{Expulsion de la désoxyhémoglobine}} .$$

L'extraction de l'oxygène est une fonction du flux, où E_0 est la fraction d'extraction en oxygène. En supposant que tout l'oxygène extrait est consommé instantanément, nous pouvons faire l'approximation

$$E(f_{entrant}) = 1 - (1 - E_0)^{\frac{1}{f_{entrant}}},$$

ce qui nous permet d'obtenir les équations d'état de ν et q,

$$\begin{aligned} \tau \frac{d\tau}{dt} &= f_{entrant}(t) - \nu^{\frac{1}{\alpha}}, \\ \tau \frac{dq}{dt} &= f_{entrant}(t)\frac{1 - (1 - E_0)^{\frac{1}{f_{entrant}}}}{E_0} - \nu(t)^{\frac{1}{\alpha}}\frac{q(t)}{\nu(t)}, \end{aligned}$$

où le flux sanguin sortant a été remplacé par sa modélisation

$$f_{sortant} = \nu^{\frac{1}{\alpha}},$$

où α est le paramètre de rigidité des vaisseaux sanguins.

Enfin, l'équation du changement du signal BOLD est exprimée comme l'équation non-linéaire

$$\lambda(q, \nu) = V_0 \left[k_1(1 - q) + k_2 \left(1 - \frac{q}{\nu}\right) + k_3(1 - \nu) \right],$$

avec

$$\begin{aligned} k_1 &= 7E_0, \\ k_2 &= 2, \\ k_3 &= 2E_0 - 0,2. \end{aligned}$$

où $V_0 = 0.02$ est la fraction du volume sanguin restante. Notons que dans cette section, nous nous sommes également inspiré de [19].

1.3 Conclusion

Dans ce chapitre, nous venons de présenter la méthode qui a permis aux médecins d'obtenir les données que nous utiliserons dans les chapitres ultérieurs : l'imagerie par résonance magnétique fonctionnelle.

Nous avons ensuite présenté le modèle que nous utiliserons afin de simuler des données obtenues par IRMf : *Dynamic Causal Modelling*. Celui-ci prend une fonction stimulus comme entrée et nous fournit le signal BOLD. C'est ce signal BOLD, Blood Oxygen Level Dependent, que fournit l'IRMf.

Dans le chapitre suivant, nous traiterons de la classification spectrale. Nous y verrons notamment les outils qui nous permettront de classifier les différents réseaux que nous avons à disposition.

Chapitre 2

Classification des réseaux

L'objectif de ce chapitre est d'introduire des outils et concepts nécessaires à la classification des réseaux. Tout d'abord nous introduisons la notion d'opérateur de Koopman qui nous sera utile pour mettre en évidence la relation qui existe entre la dynamique d'un réseau et sa structure. Nous présentons ensuite la décomposition en modes dynamiques et l'algorithme qui y est associé. Celui-ci est un outil qui permet d'analyser des systèmes dynamiques non-linéaires. Nous présentons enfin trois méthodes de classification que nous utiliserons dans les tests numériques : les arbres de décision, les Extremely Randomized Trees et la régression linéaire.

2.1 Relation entre la dynamique et le réseau

Dans cette section, nous parlerons du spectre de l'opérateur de Koopman ainsi que du spectre du réseau, et nous montrerons qu'il existe un lien entre les deux. Nous présenterons ensuite la méthode de décomposition en modes dynamiques qui nous sera utile pour avoir des informations sur le réseau considéré grâce aux modes de l'opérateur de Koopman. En effet, la décomposition en modes de Koopman est basée sur le fait que dans le cas des dynamiques non-linéaires, les modes propres sont les modes de Koopman. La théorie des modes de Koopman a été introduite dans le domaine de la mécanique des fluides, grâce à l'idée qu'une superposition de dynamiques peut décrire précisément un flot, ces dynamiques étant influencées par des caractéristiques du flot.

2.1.1 Opérateur de Koopman

Nous allons commencer par rappeler quelques concepts liés à l'opérateur de Koopman, en se basant sur les articles [24] et [2], ainsi que sur le cours [23]. L'opérateur de Koopman va d'abord être défini pour un système dynamique en temps discret, avant d'être étendu à un système dynamique en temps continu. Considérons une application non-linéaire

$$\begin{array}{rccc} T: & X & \to & X \\ & x & \rightsquigarrow & T(x) \end{array},$$

avec X un espace vectoriel de dimension finie. Nous considérons également une dynamique en temps discret

$$x_{k+1} = T(x_k), \tag{2.1}$$

$\forall k \in \mathbb{N}$. Ainsi, pour une condition initiale x_0 donnée, la position en un temps t de la particule est donnée par

$$x_t = Tx_{t-1} = T^{\circ t} x_0.$$

L'application T qui décrit l'évolution du vecteur d'état $x \in X$ n'est pas nécessairement linéaire. Dès lors, nous étudions l'évolution de x au travers d'un observable

$$\begin{array}{rccc} g: & X & \to & \mathbb{R} \\ & x & \rightsquigarrow & g(x) \end{array}$$

dans un espace fonctionnel $\mathcal{F}$, plutôt que d'étudier l'évolution de x dans l'espace X. Pour ce faire, nous devons introduire l'opérateur

$$\begin{array}{rccc} U: & \mathcal{F} & \to & \mathcal{F} \\ & g & \rightsquigarrow & Ug \end{array}$$

tel que $\forall x \in X$,

$$Ug(x) = g \circ T(x).$$

Cet opérateur U est appelé « opérateur de Koopman » , ou « opérateur de composition » , et est défini sur l'espace des fonctions observables $\mathcal{F}$. L'opérateur de Koopman a l'avantage de pouvoir traiter la dynamique considérée dans l'espace des observables et non plus dans l'espace des états, et donc d'avoir une évolution linéaire donnée par l'opérateur. Cependant, il a l'inconvénient de traiter un espace de dimension infinie. La linéarité de l'opérateur est aisément démontrable. En effet, l'opérateur de Koopman est linéaire, si $\forall \alpha_1, \alpha_2 \in \mathbb{R} \forall g_1, g_2 \in \mathcal{F}$,

$$U(\alpha_1 g_1 + \alpha_2 g_2) = \alpha_1 U(g_1) + \alpha_2 U(g_2). \tag{2.2}$$

Dès lors, nous considérons les scalaires $\alpha_1, \alpha_2 \in \mathbb{R}$ ainsi que les observables $g_1, g_2 \in \mathcal{F}$. Appliquons la définition de l'opérateur de Koopman à la première partie de l'égalité (2.2), c'est-à-dire

$$\begin{aligned} U(\alpha_1 g_1 + \alpha_2 g_2) &= (\alpha_1 g_1 + \alpha_2 g_2) \circ T \\ &= \alpha_1 g_1 \circ T + \alpha_2 g_2 \circ T, \end{aligned}$$

par la linéarité de la composition de fonctions. Et si nous appliquons à nouveau la définition de l'opérateur de Koopman, nous retombons sur l'égalité (2.2), ce qui prouve la linéarité de l'opérateur.

Intéressons-nous maintenant aux valeurs propres et fonctions propres de l'opérateur Koopman. En effet, celles-ci nous seront utiles lorsque nous présenterons la décomposition en modes de Koopman. Considérons $\phi_j : \mathbb{R}^n \to \mathbb{C}$, un observable à valeurs complexes du système dynamique

(2.1), et μ_j un nombre complexe. Le couple (ϕ_j, μ_j) est appelé « paire fonction propre - valeur propre » de l'opérateur de Koopman, s'il vérifie

$$U\phi_j = \mu_j \phi_j. \tag{2.3}$$

Cette paire fonction - valeur propre possède également une propriété intéressante qui est facilement démontrable.

Proposition 2.1.1.
Si (ϕ_i, μ_i) et (ϕ_j, μ_j) sont des paires fonction propre - valeur propre,
Alors $(\phi_i \cdot \phi_j, \mu_i \cdot \mu_j)$ l'est également.

Démonstration.
Développons la contrainte (2.3) sur la paire $(\phi_i \cdot \phi_j, \mu_i \cdot \mu_j)$.

$$\begin{aligned} U(\phi_i \cdot \phi_j) &= (\phi_i \cdot \phi_j) \circ T \\ &= (\phi_i \circ T) \cdot (\phi_j \circ T) \\ &= U\phi_i \cdot U\phi_j \\ &= (\mu_i \cdot \mu_j)\phi_i \cdot \phi_j, \end{aligned}$$

c'est-à-dire $(\phi_i \cdot \phi_j, \mu_i \cdot \mu_j)$ est une paire fonction propre - valeur propre.

□

Supposons à présent que tout observable du système dynamique (2.1) soit dans le sous-espace vectoriel engendré par les fonctions propres de Koopman, c'est-à-dire

$$g(x) = \sum_{k=0}^{\infty} g_k \phi_k(x),$$

où g_k est le coefficient associé à l'indice k. Dès lors, les observables ont une évolution définie par

$$U^t g(x) = \sum_{k=0}^{\infty} g_k \mu_k \phi_k(x), \tag{2.4}$$

ce qui signifie que l'évolution de g a un développement linéaire en termes de fonctions propres de Koopman.

Dans beaucoup de situations, les données mesurées sur un système dynamique proviennent de plusieurs observables

$$\mathbf{g} = \begin{pmatrix} g^1 \\ \dots \\ g^m \end{pmatrix},$$

où pour $1 \leqslant j \leqslant m$, $g^j : X \to \mathbb{R}$. Si nous développons chaque g^j comme en (2.4), nous obtenons le développement de $\mathbf{g} : X \to \mathbb{R}^m$, c'est-à-dire

$$U^t \mathbf{g}(x) = \sum_{k=0}^{\infty} \mathbf{g}_k \mu_k \phi_k(x).$$

Cette expression est appelée « Décomposition en modes de Koopman » de l'observable $\mathbf{g}$. Les $\mathbf{g}_k$ sont les modes de Koopman de l'observable $\mathbf{g}$ associés aux valeurs propres μ_k. Ces derniers sont une projection de l'observable sur les fonctions propres de l'opérateur de Koopman. Notons que les caractères en gras signifient que l'objet correspondant est un vecteur. Nous fixons la notation g^d pour désigner le mode dominant de Koopman, c'est-à-dire le mode associé à la valeur propre dont la partie réelle est la plus grande.

2.1.2 Algorithme de décomposition en modes dynamiques

La méthode de décomposition en modes de Koopman sera nécessaire dans notre analyse future. Cependant, nous ne l'utiliserons pas telle quelle mais nous l'approximerons par une somme finie à l'aide d'une technique de décomposition numérique appelée « décomposition en modes dynamiques », ou DMD. Avant de présenter cet algorithme, prenons un exemple inspiré de l'article [10] afin de donner une intuition de ce qu'est cette décomposition. Considérons deux musiciens sur la Place Saint-Aubain. L'un d'eux se trouve sur la gauche et joue du violoncelle. Le deuxième joue de la flûte traversière et se situe sur la droite. A ce concert s'ajoutent les bruits du trafic, des passants, etc. La musique de ce duo est enregistrée à l'aide de micros permettant de trier les données audio et situés à différents endroits. Dans les données enregistrées, nous avons une basse fréquence qui correspond au violoncelle, une haute fréquence correspondant à la flûte traversière, et quelques données de haute fréquence correspondant aux bruits externes.

Si nous appliquons la décomposition en modes dynamiques aux données, nous pouvons capturer, grâce aux modes, une basse fréquence à haute intensité dans la composante du mode associée au micro de gauche, une haute fréquence de haute intensité dans la composante du mode associée au micro de droite, et des hautes fréquences avec des intensités plus ou moins diffuses dans l'espace, et donc se retrouvant sur chacune des composantes du mode. Par conséquent, la décomposition va capturer la personne jouant du violoncelle, celle jouant de la flûte traversière, ainsi que le bruit ambiant.

L'algorithme de décomposition en modes dynamiques a été initialement développé pour une application à la mécanique des fluides. Cependant, l'algorithme s'est avéré être un outil efficace pour l'analyse de systèmes dynamiques non-linéaires, comme nous l'explique [28]. Comme nous venons de le voir, la décomposition en modes de Koopman est une décomposition en modes et valeurs propres de l'opérateur de Koopman. DMD en est son algorithme et nous nous en servirons afin d'extraire des propriétés spectrales de dynamiques non-linéaires.

La méthode de décomposition en modes dynamiques peut être présentée comme une analyse de paires de vecteurs de données (x_k, y_k) de dimension n. A partir de ces données, nous construisons un opérateur linéaire A et nous définissons DMD comme la décomposition propre de cet

opérateur.

Considérons un ensemble de paires $\{(x_k, y_k)\}_{k=1}^{m}$. Ces données peuvent être stockées en termes de matrices $n \times m$

$$X \triangleq \begin{bmatrix} | & & | \\ x_1 & \dots & x_m \\ | & & | \end{bmatrix} \quad \text{et} \quad Y \triangleq \begin{bmatrix} | & & | \\ y_1 & \dots & y_m \\ | & & | \end{bmatrix}. \tag{2.5}$$

A partir de ces deux matrices, nous pouvons alors calculer

$$A \triangleq YX^{+},$$

où X^{+} est le pseudo-inverse de X. La décomposition en modes dynamiques de la paire (X, Y) est approximée par la décomposition propre de la matrice A, c'est-à-dire les modes de Koopman et les valeurs propres de l'opérateur de Koopman sont approximées par les vecteurs propres et valeurs propres de A.

L'algorithme de décomposition en modes dynamiques que nous utiliserons est l'algorithme DMD dit « exact », qui utilise la décomposition en valeurs singulières. Nous l'utiliserons plutôt qu'une autre version car celle-ci est moins coûteuse d'après l'article [28]. Celui-ci est repris dans la table Algorithme 2.1.

Algorithme 2.1 Algorithme DMD Exact [28].

Entrées: Matrice de données $Z = [(x_1, y_1)\ ...\ (x_m, y_m)]$
Sorties: Modes DMD φ et valeurs propres DMD associées μ

1: Arranger les paires $\{(x_1, y_1),\ ..., (x_m, y_m)\}$ en matrices X et Y telles que

$$X \triangleq [x_1,\ \dots, x_m] \text{ et } Y \triangleq [y_1,\ \dots, y_m]$$

2: Calculer la factorisation en valeurs singulières (SVD) réduite de X, telle que $X = U\Sigma V^{*}$
3: Définir la matrice $\tilde{A} \triangleq U^{*}YV\Sigma^{-1}$, où U est de taille $n \times r$, V de taille $m \times r$, Σ est une matrice diagonale $r \times r$ et r est le rang de X
4: Calculer les valeurs propres et vecteurs propres de $\tilde{A}$, tels que $\tilde{A}w = \mu w$. Chaque valeur propre non nulle μ est une valeur propre DMD
5: Le mode DMD correspondant à μ est donné par

$$\varphi \triangleq \frac{1}{\mu} YV\Sigma w$$

6: Le scaling des modes est donné par

$$\varphi = \Lambda^{-1}\varphi^{-1}z_1,$$

où Λ est la matrice des valeurs propres

L'algorithme tel qu'il est défini nous fournit les valeurs propres de l'opérateur de Koopman en temps discret. Cependant, lors de son implémentation, l'algorithme nous renvoie les valeurs propres en temps continu, puisque nous prenons la valeur propre

$$\lambda = \frac{\log(\mu)}{\Delta t},$$

où Δt correspond à la période d'échantillonnage des données et μ à la valeur propre telle qu'elle apparaît dans l'Algorithme 2.1. Lorsque nous considérons le mode pré-multiplié par son scaling, appelé « mode rescaled », nous le dénoterons $\tilde{g}_i$. Nous étendrons la notation à $\tilde{g}_i^d$ lorsqu'il s'agira de la i-ème composante du mode dominant rescaled.

Nous allons à présent montrer que notre définition de DMD préserve les liens entre DMD et la théorie de l'opérateur de Koopman. Ces liens sont importants car sans eux, l'utilisation de DMD pour analyser des dynamiques non-linéaires ne serait pas justifiée. Pour ce faire, considérons les matrices définies en (2.5). Supposons que nous calculons les modes DMD des matrices X et Y, ce qui nous donne des vecteurs propres et valeurs propres qui satisfont

$$A\omega_j = \mu_j \omega_j,$$

où $A = YX^+$. Si la matrice A a un ensemble complet de vecteurs propres, nous pouvons exprimer chaque colonne x_k de X comme

$$x_k = \sum_{j=0}^{n-1} c_{jk}\varphi_j,$$

pour certaines constantes c_{jk}. Pour des données linéairement consistantes, nous avons $Ax_k = y_k$, et donc

$$\begin{aligned} y_k &= Ax_k \\ &= \sum_{j=0}^{n-1} c_{jk} A\varphi_j \\ &= \sum_{j=0}^{n-1} \mu_j c_{jk}\varphi_j. \end{aligned}$$

Nous pouvons dès lors comparer ces résultats à l'équation (2.4). Nous observons que dans le cas de matrices de données linéairement consistantes, les modes DMD correspondent aux modes de Koopman, et les valeurs propres DMD correspondent aux valeurs propres de Koopman, où les constantes c_{jk} sont données par $c_{jk} = \mu_k$.

2.1.3 Spectres de l'opérateur de Koopman et du réseau

Gardons en mémoire l'objectif premier de ce mémoire : classer des réseaux sur base de propriétés spectrales. Maintenant que nous avons introduit les différents concepts liés à l'opérateur de Koopman, nous aimerions pouvoir déduire la structure d'un réseau considéré. Pour cela, basons-nous sur les résultats de l'article [22]. La méthode présentée, l'identification spectrale des réseaux, a été développée dans le but de fournir la connectivité du réseau sous-jacent, avec une contrainte importante : nous ne connaissons que partiellement la dynamique des éléments individuels.

La motivation principale de cette méthode est que les méthodes d'identification des réseaux développées jusqu'à présent imposent la mesure d'information en tout nœud. Ce n'est cependant ni réalisable, ni réaliste. En effet, les réseaux réels ont généralement beaucoup plus de nœuds qu'il n'y a de capteurs disponibles à la mesure. De plus, certains nœuds sont tout simplement inaccessibles. Dès lors, nous avons dans une majorité des cas des mesures effectuées sur un ensemble de nœuds et non pas sur des nœuds isolés. Dans la deuxième partie de ce travail, nous serons dans le cas sus-cité, c'est-à-dire que nous n'aurons de l'information que pour un groupe de nœuds (un groupe de neurones). De plus, nous nous concentrerons sur l'identification des propriétés spectrales du réseau, puisque nous ne nous intéresserons pas aux relations inter-neuronales au sein des différents groupes de neurones (les régions).

Les propriétés spectrales d'un réseau sont définies par la matrice Laplacienne associée au réseau. Ces propriétés peuvent nous donner des informations intéressantes sur la structure globale du réseau, telles que le degré du nœud moyen, le degré du nœud minimum, le degré du nœud maximum, et la connectivité du réseau. Prenons par exemple la deuxième plus petite valeur propre de la matrice Laplacienne, aussi appelée connectivité algébrique. Elle est liée à la vitesse de la diffusion de l'information et joue un rôle dans l'étude de la synchronisation du réseau [22]. De manière générale, les propriétés spectrales donnent des indications quant à la structure globale du réseau. Nous pouvons donc utiliser ces indicateurs spectraux afin de comparer différents réseaux.

Nous pouvons extraire ces propriétés spectrales des dynamiques non-linéaires grâce à diverses méthodes numériques, et notamment grâce à l'algorithme de décomposition en modes dynamiques, que nous avons décrit dans la section 2.1.2. Une fois ces propriétés spectrales de la dynamique du réseau extraites, il nous est possible de les lier aux propriétés spectrales du réseau.

2.2 Classification spectrale

Comme nous l'avons vu dans les points précédents, travailler sur le spectre de la dynamique du réseau est équivalent à travailler sur le spectre du réseau lui-même. Le but de ce travail étant de pouvoir déterminer l'état de conscience d'un patient, état qui correspond à un type de réseau neuronal particulier, il nous faudra faire une classification sur base d'éléments spectraux, comme évoqué précédemment. Pour ce faire, nous utiliserons trois méthodes de classification supervisée : les arbres de décision, les Extremely Randomized Trees et la régression logistique. Cependant, lorsque nous utiliserons ces méthodes, nous porterons un plus grand intérêt aux *Extremely Randomized Trees*. Nous utiliserons les deux autres méthodes afin de comparer nos résultats.

Les trois méthodes sont des techniques de classification supervisée. Le but de la classification supervisée est de définir des règles qui permettent, en se basant sur des variables caractéristiques, de classer des objets dans les classes adéquates. Le terme « supervisé » fait référence au fait que nous fournissons à la méthode les caractéristiques sur lesquelles nous voulons effectuer la classification, mais nous lui donnons également les classes auxquelles les données appartiennent. Concrètement, nous disposons d'un ensemble de données dont les classes sont connues. Cet ensemble est subdivisé en un ensemble dit « d'apprentissage » qui nous sert à entraîner les méthodes de classification, et un ensemble dit « de test » que nous utilisons pour étudier la fiabilité des règles de décision établies par le premier sous-ensemble. Une fois le modèle (c'est-à-dire les règles de décision) validé, nous serons capable de prédire la classe d'un nouvel objet avec une certitude établie lors des différents tests de fiabilité.

Nous pouvons écrire ce principe plus formellement, de manière similaire à l'article [3]. Nous avons n réalisations supposées indépendantes

$$(x_i, y_i)_{1 \leqslant i \leqslant n} \subset \mathcal{X} \times \mathcal{Y}$$

d'une variable aléatoire (X, Y) suivant une loi inconnue P. La variable X est un ensemble de caractéristiques observables d'un objet, tandis que la variable Y représente une caractéristique inobservable et est appelée « étiquette associée à X ». Dans le cas d'une situation appliquée au milieu médical, X est un ensemble de caractéristiques du patient et Y donne une information sur son état de conscience. L'objectif de la classification supervisée est de prédire y_{n+1} grâce à x_{n+1} et $(x_i, y_i)_{1 \leqslant i \leqslant n}$ avec un taux d'erreur aussi faible que possible. La paire (x_{n+1}, y_{n+1}) est une nouvelle réalisation de (X, Y) et est indépendante de $(x_i, y_i)_{1 \leqslant i \leqslant n}$.

Afin d'obtenir les meilleures classifications possibles, nous appliquons la méthode dite de *cross-validation*. Cette méthode permet d'éviter au maximum le sur-apprentissage d'un modèle. En effet, lors de son apprentissage, le modèle essaye de coller au mieux aux données. Ainsi, lorsque nous prenons un ensemble de données indépendant de celles utilisées pour l'apprentissage, il se peut que la prédiction soit très mauvaise puisque le modèle s'est trop adapté aux premières données. Dès lors, la *cross-validation* va effectuer plusieurs ajustements du modèle. En effet, nous subdivisons l'ensemble de données en n sous-ensembles. Un de ces sous-ensembles constitue les données test, tandis que les autres seront utilisés pour l'entrainement du modèle. Ensuite, nous sélectionnons un autre de ces sous-ensembles et effectuons à nouveau l'entrainement du modèle. Nous répétons ce processus au total n fois. Dans notre cas, un sous-ensemble correspond aux données d'un patient dans ses différents états de conscience. Ainsi, lors de la première itération, l'ensemble test sera constitué des données du premier patient.

Nous allons à présent présenter les trois méthodes de classification que nous utiliserons dans la suite de ce travail, à savoir les arbres de décision, les Extremely Randomized Trees et la régression logistique.

2.2.1 Arbres de décision

Nous baserons cette introduction aux arbres de décision sur le cours [12]. Les arbres de décision sont une méthode de classification supervisée. L'idée derrière un arbre de décision est que

si une personne veut prendre une décision, elle va suivre une série de règles pour y arriver. Cette manière de raisonner est propre aux humains et fortement utilisée dans le milieu médical. Nous pouvons prendre des décisions sur base de règles dans tout domaine. Par exemple, dans le cas d'un commercial chez un opérateur téléphonique, afin de savoir quel type d'abonnement vendre à un client, le commercial pourrait avoir les règles

— `if (client_age < 23)` $\wedge$ `(has_car=false) then product = voice_3G`,
— `if (client_age > 65)` $\wedge$ `(has_car=true) then product = voice_only`,
— `if (client_age < 15)` $\wedge$ `(prepaid=true) then product = text_only`.

Cependant, cette manière d'écrire les règles est presque illisible, surtout lorsque nous arrivons à un système de diagnostic plus complexe. C'est également problématique dans le cas où les règles sont obtenues à partir de données. Les arbres de décision résolvent ce problème de lisibilité et permettent même d'ajouter des règles aisément. Un arbre de décision est composé de nœuds, qui peuvent se diviser en trois catégories :

— Le nœud racine : la lecture de l'arbre commence par ce nœud et il n'a, par conséquent, aucune arête incidente,
— Les nœud internes : ils possèdent une arête incidente et au moins deux arêtes sortantes qui mènent vers d'autres nœuds,
— Les feuille, ou nœuds terminaux : ils possèdent une arête incidente mais ne possèdent pas de descendants.

La construction d'un arbre de décision se fait de manière récursive et a pour but de produire des groupes d'individus les plus homogènes possibles pour chaque classe. En effet, à chaque nœud, un critère de coupure est sélectionné afin de séparer l'ensemble des données. Typiquement, ce critère prend la forme

$$x_i \leq c_i,$$

où c_i est le seuil de coupure et x_i est une des caractéristiques d'entrée sélectionnée de manière à maximiser la diminution de l'impureté du nœud. Le choix de l'impureté dépend de l'algorithme utilisé. Par exemple, l'algorithme CART, utilisé par défaut dans Matlab, utilise l'indice de Gini, qui mesure la dispersion d'une distribution, pour quantifier cette impureté. Le critère d'arrêt par défaut des arbres est que chaque feuille ne contienne plus que des données pures, c'est-à-dire appartenant à une même classe.

La lecture d'un arbre de décision commence toujours au nœud racine et se poursuit jusqu'aux feuilles. La racine et chaque nœud interne de l'arbre correspondent chacun à une règle de décision. Les arêtes sortantes sont quant à elles associées aux valeurs prises par la règle du nœud parent. Les feuilles de l'arbre sont associées à une décision. Lorsque l'arbre de décision est utilisé pour une classification, les feuilles correspondent aux classes potentiellement attribuables aux objets à classer.

Prenons un simple exemple pour illustrer la manière d'utiliser un arbre de décision. Prenons le diagnostic de la dengue, une infection virale et endémique des pays tropicaux. Le but de l'arbre de décision est de savoir si un patient est atteint de la dengue ou non. Pour cela, nous avons à disposition trois tests, que nous utiliserons comme les caractéristiques de division d'un nœud : le taux de lymphocytes, de plaquettes, et de globules blancs. Les valeurs que peuvent prendre ces caractéristiques sont « haut » ou « bas ». L'arbre de décisions correspondant à ce cas se trouve à la FIGURE 2.1.

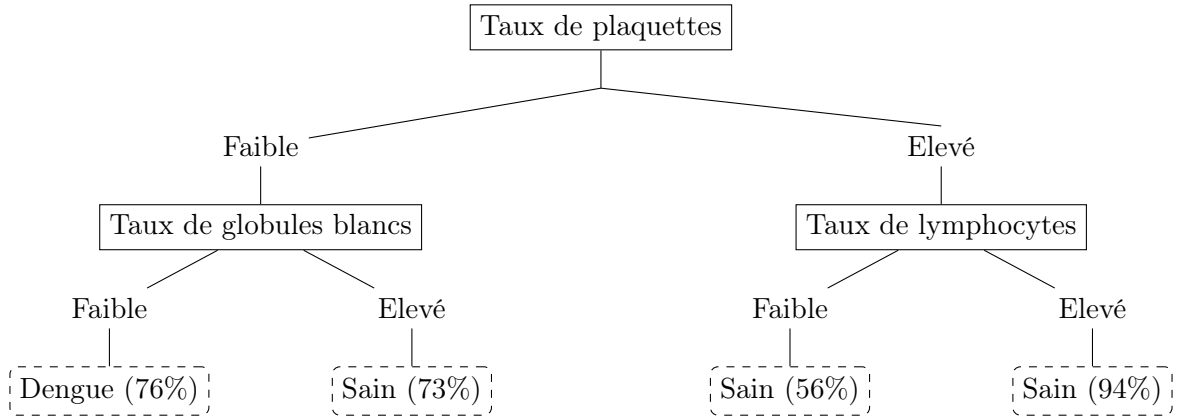


FIGURE 2.1 – Arbre de décision appliqué au diagnostique de la dengue.

Remarquons que selon le chemin emprunté, nous ne nous posons pas les mêmes questions. En effet, si le patient a un faible taux de plaquettes, nous vérifions son taux de globules blancs, tandis que si le taux de plaquettes est élevé, nous regardons le taux de lymphocytes. Prenons la feuille la plus à droite. Le patient est sain. Le pourcentage associé signifie que parmi la population utilisée pour construire l'arbre de décision, 94% des individus possédant un haut taux de plaquettes et un haut taux de lymphocytes étaient bel et bien sains. Nous pouvons par conséquent dire que un patient arrivant dans cette feuille est sain, avec un degré de confiance élevé. Ce degré de confiance diminue si nous prenons la feuille juste à gauche. Notons qu'il s'agit ici d'un arbre de décision binaire, c'est-à-dire que nous n'avons que deux classes possibles. Dans notre exemple, ces deux classes sont « malade » et « sain » . Dans la suite de ce travail, nous serons amenés à construire des arbres non-binaires, c'est-à-dire avec plus de deux classes cibles.

2.2.2 Extremely Randomized Trees

Le principe des *Extremely Randomized Trees* est de créer plusieurs arbres de décision, c'est-à-dire une forêt d'arbres. Cependant, la création de ces arbres ne s'effectue pas de la même manière que pour les arbres de décision de la section précédente. Dans la création des arbres de cette méthode, de l'aléatoire est introduit dans la manière de diviser les feuilles : les critères de coupures sont sélectionnés parmi un sous-ensemble extrait aléatoirement de l'ensemble des caractéristiques.

Les *Extremely Randomized Trees* font partie des méthodes de classification basées sur les arbres de décision aléatoires. En effet, ces méthodes introduisent de l'aléatoire dans leur proces-

sus de partitionnement, ou dans le choix des caractéristiques utilisées comme critère de coupure. L'introduction de l'aléatoire a pour but de produire un ensemble de modèles plus ou moins diversifiés dont les prédictions sont agrégées afin d'obtenir la prédiction finale.

L'algorithme Extra-Trees (pour Extremely Randomized Trees) construit un ensemble d'arbres de décisions. Comme nous venons de le signaler, la principale différence avec les autres méthodes basées sur les arbres réside dans la sélection des caractéristiques pour la division d'un nœud, mais elle se distingue également dans l'entrainement des arbres. En effet, elle n'utilise pas de bootstrap de l'ensemble d'apprentissage, c'est-à-dire qu'elle utilise tout l'ensemble de données d'apprentissage sans effectuer de rééchantillonnage pour chaque arbre.

Une fois les différents arbres créés, les prédictions de ceux-ci sont agrégées afin d'obtenir la prédiction finale. L'agrégation se fait par vote majoritaire, c'est-à-dire que la prédiction qui aura été prédite le plus de fois correspondra à la prédiction finale.

L'algorithme Extra-Trees possède deux paramètres importants. Le premier, K, détermine le nombre de caractéristiques tirées aléatoirement afin d'effectuer les subdivisions des nœuds. Celui-ci peut être choisi dans l'intervalle $[1, \dots, n]$, où n est le nombre de caractéristiques disponibles. Pour un problème donné, plus la valeur de K est petite, plus les arbres sont aléatoires et plus faible est la dépendance de ceux-ci sur les classes de l'ensemble d'entrainement. Nous pouvons avoir deux cas extrêmes dûs au choix de K :

— $K = 1$: les coupures des nœuds se font indépendamment des variables d'entrainement (nous sommes dans le cas de Totally Randomized Trees),
— $K = n$: le choix des caractéristiques n'est plus explicitement aléatoire et l'effet de la randomisation agit uniquement sur le choix des points de coupure.

Le second paramètre, M, correspond au nombre d'arbres que l'algorithme va créer. Pour les méthodes de randomisation, l'erreur de prédiction est une fonction de M, monotone décroissante. Par conséquent, plus M est élevé, plus la méthode sera précise.

Grâce à la division de l'ensemble de données en un ensemble d'entrainement et de test, ainsi que grâce à l'introduction de l'aléatoire, notamment lors de la sélection des caractéristiques pour la création des arbres, la méthode Extra-Trees évite l'overfitting, c'est-à-dire qu'elle évite que les arbres créés ne collent trop aux données. C'est un avantage indéniable car cela nous permet d'assurer que si nous appliquons la méthode afin de prédire la classe de données issues d'un autre jeu de données que le nôtre, celle-ci sera capable de prédire la bonne classe, avec une précision calculée lors de la phase test.

Choix des paramètres

Le choix des paramètres de la méthode sont discutés dans l'article [14]. Nous prendrons par conséquent les valeurs de paramètres qui semblent les plus optimales, c'est-à-dire

— $K = \sqrt{n}$, où n est le nombre de caractéristiques disponibles,
— $M = 100$.

Pseudo-code de l'algorithme Extra-Trees

Comme nous venons de le voir, cette méthode de classification introduit de l'aléatoire à plusieurs endroits. Nous savons également que la construction d'un arbre de décision se fait de manière récursive. L'ALGORITHME 2.2 reprend le pseudo-code de la méthode des Extra-Trees.

Nous pouvons voir que trois procédures suffisent à créer les forêts aléatoires que nous considérons, et que l'aléatoire de la méthode se situe au niveau de la procédure de sélection de la règle de coupure PICK_A_RANDOM_SPLIT.

2.2.3 Régression logistique

La régression logistique est un modèle de classification linéaire [8] qui permet de classer des observations caractérisées par des variables $\{x_i\}_{i=1,\dots n}$ continues réelles. Dans notre cas, la variable cible ne peut prendre que deux valeurs, comme montré sur la FIGURE 2.2, c'est pourquoi nous ne pouvons pas utiliser la régression linéaire pour discriminer ces variables. Pour créer un modèle de régression logistique, nous devons commencer par créer une fonction de régression.

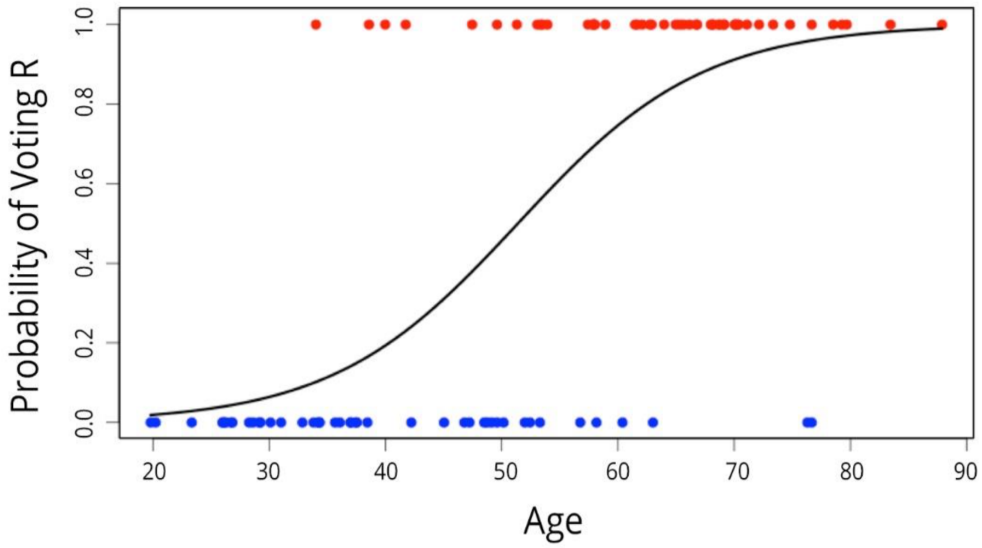


FIGURE 2.2 – La variable cible, Y, ne prend que deux valeurs, c'est pourquoi nous ne pouvons pas utiliser une fonction de régression linéaire. Nous utilisons donc la fonction logistique, plus adaptée pour ce cas-ci. Source : [4].

Considérons dans un premier temps une classification binaire. Nous faisons l'hypothèse que la probabilité d'appartenir à la classe « positif », $P(Y = 1|X = x)$, est une fonction logistique, avec un score linéaire. Le score est donc défini comme une régression linéaire des caractéristiques considérées, c'est-à-dire

$$S(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n = \beta_0 + \sum_{i=1}^{n} \beta_i x_i \in \mathbb{R},$$

Algorithme 2.2 Pseudo-code de l'algorithme Extra-Trees [14].

Entrées: Un ensemble d'entrainement S
Sorties: Un ensemble d'arbres $\mathcal{T} = \{t_1, \dots t_M\}$

procedure BUILD_AN_EXTRA_TREE_ENSEMBLE(S)
  for $i = 1 : M$ **do**
    Générer un arbre : $t_i =$ BUILD_AN_EXTRA_TREE(S)
  end for
  return $\mathcal{T}$
end procedure

Entrées: Un ensemble d'entrainement S
Sorties: Un arbre t

procedure BUILD_AN_EXTRA_TREE(S)
  if $|S| < n_{min}$ || toutes les caractéristiques candidates sont constantes dans S || la variable de sortie est constante dans S **then**
    return une feuille labellisée par les fréquences de classe
  else
    Sélectionner de manière aléatoire K caractéristiques $a_1, \dots, a_K$ sans remise, parmi toutes les caractéristiques candidates non constantes dans S
    Générer K partitionnements, où $s_i =$ PICK_A_RANDOM_SPLIT(S, a_i), $\forall i = 1, \dots, K$
    Sélectionner une coupure s_* telle que $Score(s_*, S) = \max_{i=1,\dots,K} Score(s_i, S)$
    Partitionner S en deux sous-ensembles S_l et S_r en se basant sur s_*
    Construire $t_l =$ BUILD_AN_EXTRA_TREE(S_l) et $t_r =$ BUILD_AN_EXTRA_TREE(S_r) sur base de ces deux sous-ensembles
    Créer un nœud avec la coupure s_* et lier les sous-arbres S_l et S_r à ce nœud
    return L'arbre t
  end if
end procedure

Entrées: Un ensemble d'entrainement S et une caractéristique a
Sorties: Une règle de coupure

procedure PICK_A_RANDOM_SPLIT(S, a)
  if La caractéristique a est numérique **then**
    Calculer la valeur maximale et minimale de a dans S, notées a^S_{min} et a^S_{max}
    Tirer aléatoirement un point de coupure a_c dans $[a^S_{min}, a^S_{max}]$
    return La règle de coupure « $a < a_c$ »
  end if
  if Si la caractéristique a est catégorique (où nous notons $\mathcal{A}$ son ensemble de valeurs possible) **then**
    Calculer $\mathcal{A}_S$ le sous ensemble de $\mathcal{A}$ des valeurs de a qui apparaissent dans S
    Tirer aléatoirement un sous ensemble propre non-vide $\mathcal{A}_1$ de $\mathcal{A}$ et un sous-ensemble $\mathcal{A}_2$ de $\mathcal{A} \backslash \mathcal{A}_S$
    return La règle de décision « $a \in \mathcal{A}_1 \cup \mathcal{A}_2$ »
  end if
end procedure

où les x_i sont les variables prédictives. La fonction logistique est quant à elle définie par

$$f(x) = \frac{1}{1+e^{-x}}.$$

Pour obtenir la probabilité de succès, c'est-à-dire $P(Y = 1|X = x)$, nous devons appliquer la fonction logistique à la fonction score, c'est-à-dire que la probabilité de succès est donnée par

$$p \overset{\text{déf}}{=} P(Y = 1|X = x) = \frac{1}{1 + e^{-\left(\beta_0 + \sum\limits_{i=1}^{n} \beta_i x_i\right)}},$$

où les paramètres β_i sont inconnus et estimés grâce au maximum de vraisemblance. Une fois ceux-ci estimés, nous avons une règle de classification h qui affecte à une observation x la classe 1, si $p \geq 0,5$ et à la classe 0 sinon.

Nous pouvons étendre la régression logistique binaire aux cas multi-classes. L'idée est de faire du *one-versus-all*, c'est-à-dire que nous subdivisons le problème de classification en plusieurs sous-problèmes de classification binaire. Cette méthode est représentée sur la FIGURE 2.3. Nous avons ainsi autant de fonctions de prédictions h^i que de classes considérées. Celles-ci nous donnent la probabilité qu'une observation x appartienne à la classe Y^i. Nous pourrons donc aisément savoir à quelle classe appartient l'observation x. En effet, ce sera la classe pour laquelle la fonction de prédiction associée nous donne la plus grande probabilité, c'est-à-dire que la classe de x est

$$Y = arg \max_{i \in C} h^i(x),$$

où C est l'ensemble des classes considérées.

2.2.4 Mesures de la qualité de la classification

Lorsqu'un test est appliqué à un problème du monde médical, il existe deux erreurs évidentes : il pourrait ne pas détecter une personne atteinte de la maladie sensée être détectée par ce test, ou il pourrait classer une personne saine comme étant atteinte de la maladie. Dans certains cas, une erreur est plus importante que l'autre. C'est pour minimiser au maximum les erreurs que des mesures de qualité ont été développées.

Nous présentons à présent les deux mesures de qualité que nous utiliserons afin de valider nos classifications : la matrice de confusion et la courbe ROC. Il est cependant nécessaire de commencer par présenter certains termes relatifs à la sensibilité et à la spécificité de la classification. Ces définitions sont en partie inspirées de la référence [25].

Considérons un classifieur binaire, c'est-à-dire que la méthode de classification prédit deux classes distinctes. Afin de mieux comprendre ces notions, prenons un exemple simple tiré de [11]. Nous nous plaçons dans le cadre de la médecine et nous considérons un patient. Le test a pour but de prédire si le patient est atteint d'une maladie quelconque. Nous aurons donc une classe dite « malade » et une classe dite « saine » . Nous distinguons quatre types de résultats. Ceux-ci

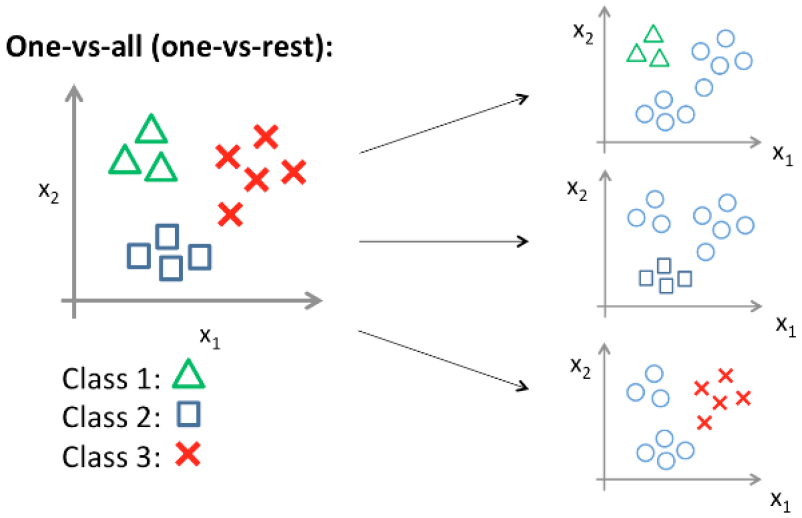


FIGURE 2.3 – Illustration de l'utilisation du « one-versus-all » pour la régression logistique. Dans ce problème, nous avons trois classes pour nos données. Dans un premier temps, nous effectuons une régression logistique afin de classer les observations de la classe 1. Nous considérons donc les deux autres classes comme étant une seule et même classe « négatif ». Une fois la fonction de prédiction calculée pour cette classe, nous faisons de même pour la deuxième puis pour la troisième classe. Une nouvelle observation appartiendra à la classe pour laquelle la fonction de prédiction renvoie la probabilité la plus élevée. Source : [5].

sont définis ci-dessous et représentés à la FIGURE 2.4.

— Vrais positifs (TP) : individus malades dont le test est positif,
— Faux positifs (FP) : individus sains dont le test est positif,
— Faux négatifs (FN) : individus malades dont le test est négatif,
— Vrais négatifs (TN) : individus sains dont le test est négatif.

Les différents résultats que nous pouvons avoir lors d'un test sont repris dans la TABLE 2.1.

	Malade	Sain
Test positif	TP	FP
Test négatif	FN	TN

TABLE 2.1 – Résultats possibles lors de la mesure de validité d'un test. TP : vrais positifs, FP : faux positifs, FN : faux négatifs, TN : vrais négatifs.

La précision d'une classe correspond à la proportion d'individus appartenant réellement à cette classe, parmi ceux prédits dans cette classe par la méthode. Nous pouvons retrouver cette

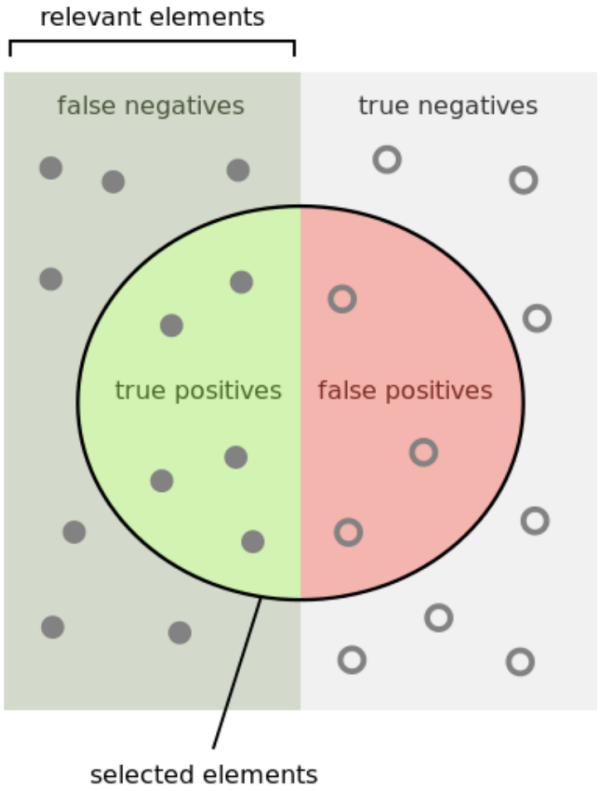


FIGURE 2.4 – Schéma représentant les données considérées ainsi que la classe « positif » dans laquelle certaines ont été regroupées (dans l'ellipse centrale). Les éléments qui ne sont pas dans cette classe sont dans la classe « négatif ». Les différentes possibilités concernant les vrais et faux positifs et négatifs sont également représentées. Nous pouvons ainsi comprendre de manière visuelle chaque notion. Source : [29].

valeur grâce à la formule

$$P = \frac{TP}{TP + FP}.$$

La sensibilité d'un test est sa capacité à donner un résultat positif lorsqu'une hypothèse est vérifiée, c'est-à-dire que la sensibilité Se, également appelée « recall », nous donne la proportion de résultats réellement positifs qui a été correctement identifiée. La spécificité Sp, quant à elle, mesure la capacité d'un test à donner un résultat négatif lorsque l'hypothèse n'est pas vérifiée. Autrement dit, la sensibilité du test considéré ci-dessus est sa capacité à prédire que le patient est malade, sachant qu'il est effectivement malade. La spécificité est la probabilité que le patient soit dit sain, lorsqu'il l'est. Nous pouvons exprimer ces deux mesures à l'aide des notions vues ci-dessus.

$$\begin{aligned} Se &= \frac{TP}{TP+FN} \\ Sp &= \frac{TN}{TN+FP} \end{aligned}$$

Nous pouvons également nous demander si un test doit plutôt être spécifique ou sensible. Dans le meilleur des mondes, un test est à la fois spécifique et sensible. Cependant, peu de tests le sont. Nous devrons donc sélectionner une tendance pour l'une ou l'autre caractéristique selon la situation à laquelle nous serons confrontés. En effet, plus un test est sensible, moins il aura de faux négatifs, et donc plus il sera capable d'exclure la maladie s'il est positif. Par contre, plus un test est spécifique, moins il aura de faux positifs et donc mieux il confirmera la maladie s'il est positif.

Par conséquent, il est préférable d'avoir un test très sensible lorsque nous voulons faire du dépistage. En effet, le but du dépistage est de détecter le plus grand nombre possible de patients atteints par la maladie. Par contre, il nous faudra un test le plus spécifique possible lorsque nous avons des patients dépistés positivement puisque notre but sera alors de minimiser les faux positifs.

Nous pouvons calculer deux autres valeurs qui peuvent être intéressantes : la valeur prédictive positive, également appelée précision, et la valeur prédictive négative. Cette première correspond à la probabilité d'être malade lorsque le test est positif, tandis que la seconde correspond à la probabilité d'être sain lorsque le test est négatif. Ces deux valeurs sont particulièrement intéressantes dans le cas où le médecin est confronté aux résultats du test effectué sur un patient. En effet, il doit savoir si le patient a effectivement la maladie ou non. Ainsi, si le test est positif, il doit connaître la probabilité que son patient soit effectivement malade, tout comme si celui-ci est négatif, la probabilité que le patient soit effectivement sain.

Lorsque nous avons plusieurs classes à prédire, la sensibilité du test correspond à la moyenne des sensibilités correspondant à chacune des classes. Notons que lorsque les différentes classes d'une classification multi-classes possèdent toutes le même nombre d'individus, la sensibilité du test correspond à l'*accuracy*.

L'*accuracy* permet de caractériser la proportion de vrais résultats prédits par la méthode. La formule de l'*accuracy* est donnée par l'égalité

$$A = \frac{TP+TN}{TP+FP+TN+FN}.$$

Chaque méthode de classification dispose d'un ou de plusieurs seuils afin de décider si une valeur prédite est considérée comme positive ou négative. La valeur de ce seuil influence la sensibilité et la spécificité du test, et donc de ses valeurs prédictives. Mais nous verrons ce seuil de manière plus concrète lorsque nous parlerons de la courbe ROC.

Nous reprenons dans la Table 2.2 les différentes notions importantes concernant la qualité de la classification, ainsi que leurs formules respectives.

Mesure de qualité	Formule
Sensibilité	$Se = \frac{TP}{TP + FN}$
Spécificité	$Sp = \frac{TN}{TN + FP}$
Précision	$P = \frac{TP}{TP + FP}$
Accuracy	$A = \frac{TP + TN}{TP + FP + TN + FN}$

TABLE 2.2 – Table reprenant les notions et formules de qualité de la classification.

Matrice de confusion

Une matrice de confusion est un outil qui permet de mesurer la qualité d'un système de classification. La représentation de celle-ci peut prendre deux orientations de lecture. Nous prenons cependant la convention suivante : les colonnes de la matrice de confusion contiennent le nombre d'occurrences d'une classe réelle, tandis que sur les lignes, nous y retrouvons le nombre d'occurrences d'une classe prédite. Si nous reprenons le cas d'un classifieur binaire, nous retrouvons la TABLE 2.1.

La diagonale de la matrice de confusion correspond aux observations qui sont correctement prédites, c'est-à-dire les vrais positifs et les vrais négatifs. Partout ailleurs, les observations sont mal prédites, et correspondent aux faux négatifs pour la colonne de la classe considérée et aux faux positifs pour la ligne de cette classe. Lorsque nous utiliserons les matrices de confusion, nous aurons également une autre information : le nombre d'observations et le pourcentage du nombre total d'observations dans chaque cellule.

Sur la FIGURE 2.5, nous pouvons observer une ligne et une colonne grisées. Dans la colonne, nous avons le pourcentage de bonnes et de mauvaises classifications pour chaque classe réelle. Le pourcentage de bonnes prédictions, qui correspond à la probabilité d'avoir une bonne prédiction et donc intéresse le médecin, correspond à la précision tandis que le pourcentage de mauvaises prédictions est appelé le taux de fausses découvertes. Dans la ligne grisée, nous avons le pourcentage de bonnes et mauvaises classifications pour chaque classe prédite. Le pourcentage de bonnes prédictions correspond à la sensibilité de la classification, comme décrit dans la section précédente, tandis que le pourcentage de mauvaises prédictions est appelé le taux de faux négatifs. L'*accuracy* du classifieur est donnée dans la cellule du coin inférieur droit.

La matrice de confusion peut-être étendue à une classification à plus de deux classes, cependant elle ne correspondra plus directement à la TABLE 2.1. En effet, pour retrouver les mêmes dénominations, il nous faudra agréger les différentes classes. Ainsi, pour avoir les TP pour une

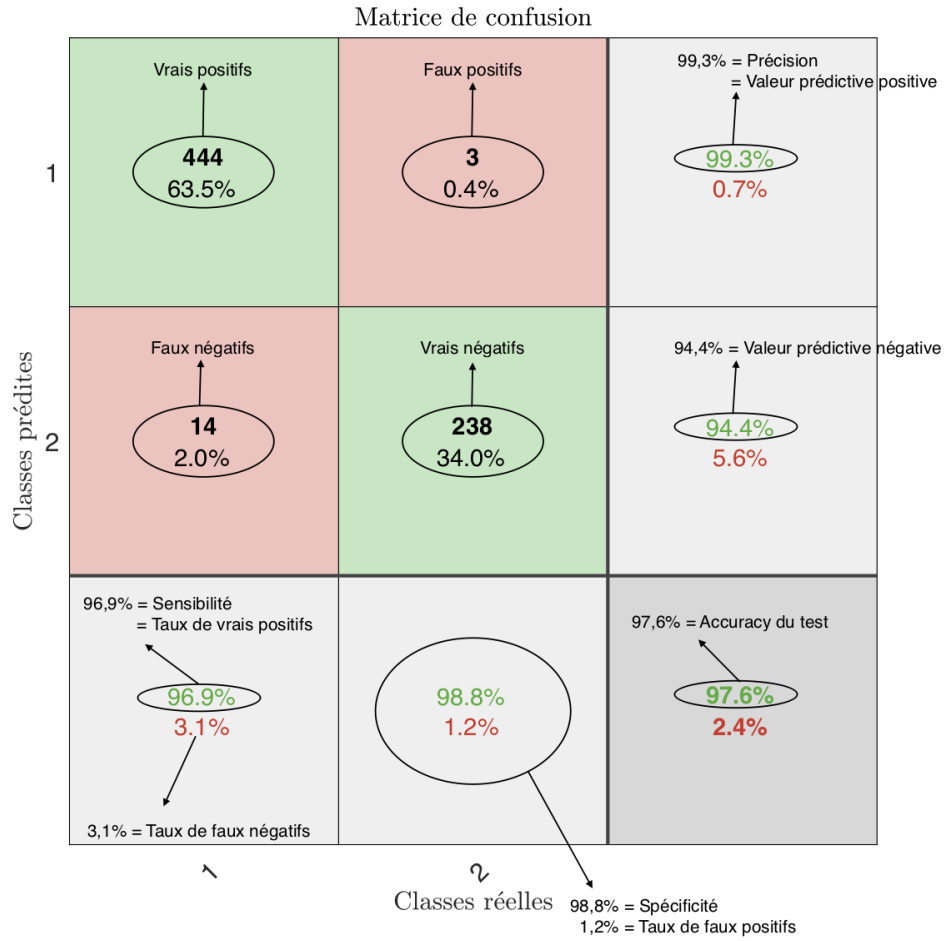


FIGURE 2.5 – Matrice de confusion pour un exemple donné dans l'aide Matlab [20]. La classe considérée est la classe 1, associée à la valeur « Positif ». La classe 2 est quant à elle associée à la classe « Négatif ».

classe donnée, nous considérerons cette classe en tant que telle, et toutes les autres comme une seule et même autre classe. Notons tout de même qu'une matrice de confusion à plus de deux classes est utilisable de la même manière qu'une matrice de confusion pour un classifieur binaire. La matrice de confusion peut être également normalisée. Dans ce cas, au plus elle se rapprochera d'une matrice identité, au mieux sera le système de classification correspondant.

Courbe ROC

La courbe ROC, pour *Receiving Operator Characteristic*, est également utilisée afin de mesurer la performance d'un classifieur. Son principal avantage est que la performance qu'elle indique est indépendante de la population de chaque classe.

Afin d'expliquer au mieux à quoi correspond la courbe ROC, commençons par définir ce qu'est le score d'une méthode. Pour ce faire, nous nous plaçons à nouveau dans le cas d'un classifieur binaire. La classe à laquelle nous nous intéressons est la classe « 1 ». Le score associé à l'entrée x correspond à la probabilité

$$s = P(Y = 1|X = x).$$

Pour définir une règle de classification à partir du score, il faut fixer un seuil. Par exemple, x appartiendra à la classe 1 lorsque

$$s \geqslant 0,5.$$

Si nous modifions ce seuil, nous modifions également la règle de classification, mais également la matrice de confusion et tous les indicateurs vrais positifs et autres. Pour construire la courbe ROC, il faut calculer le taux de vrais positifs et de faux positifs, c'est-à-dire les éléments de lignes de la matrice de confusion pour la classe considérée, en fonction d'une série de seuils. Ainsi, nous obtenons une courbe comme à la FIGURE 2.6, sur laquelle nous avons la sensibilité du test, en fonction de la quantité « $(1 - Sp)$ ».

Considérons les cas extrêmes concernant les seuils. Lorsque nous prenons le seuil le plus grand possible, nous nous retrouvons dans le coin inférieur gauche de la courbe ROC. En effet, pour le plus grand seuil, toutes les prédictions sont négatives, et nous n'avons donc aucune prédiction qui peut être vrai ou faux positif. Dans le cas contraire, lorsque nous prenons le plus petit seuil possible, nous nous trouvons dans le coin supérieur droit du graphe, car toutes les prédictions sont au dessus de ce seuil, et par conséquent, prédites positives.

Lorsque cette courbe coïncide avec la diagonale, le score de la méthode n'est pas plus performant qu'un classifieur aléatoire. En effet, cela signifie que la classe est attribuée aléatoirement à chaque observation. Le modèle sera par contre d'autant meilleur que sa courbe ROC se rapproche du coin supérieur gauche. Dans ce cas, il y a plus de vrais positifs possibles que de faux positifs.

L'aire sous la courbe ROC peut également servir de mesure de la qualité du score et donc de la classification. Elle est nommée AUC, pour *Area Under the Curve*. Nous pouvons nous en servir également pour comparer plusieurs méthodes entre elles. Sa valeur varie entre 0, pour le pire des cas, et 1 pour un classifieur parfait, en passant par 0.5 lorsque celui-ci est purement aléatoire.

Selon le type de test que nous voulons avoir, il sera préférable de se trouver dans une zone du graphe plutôt qu'une autre. Par exemple, pour un test de dépistage d'un cancer, il est préférable d'avoir un haut taux de vrais positifs avec en contre-partie un haut taux de faux positifs puisque le but d'un test de dépistage est d'identifier tous les cas potentiels. Dès lors, il sera préférable de se trouver dans la partie supérieure droite et donc d'avoir un seuil relativement bas. Par contre, dans le cas d'un test grâce auquel nous voudrions confirmer le potentiel cancer, il nous faut minimiser les faux négatifs et donc se retrouver dans la partie supérieure gauche du graphe. Concrètement, quel que soit le type de test que nous voudrions avoir, il sera essentiellement question de compromis entre la sensibilité et la spécificité, ou entre la sensibilité et la précision.

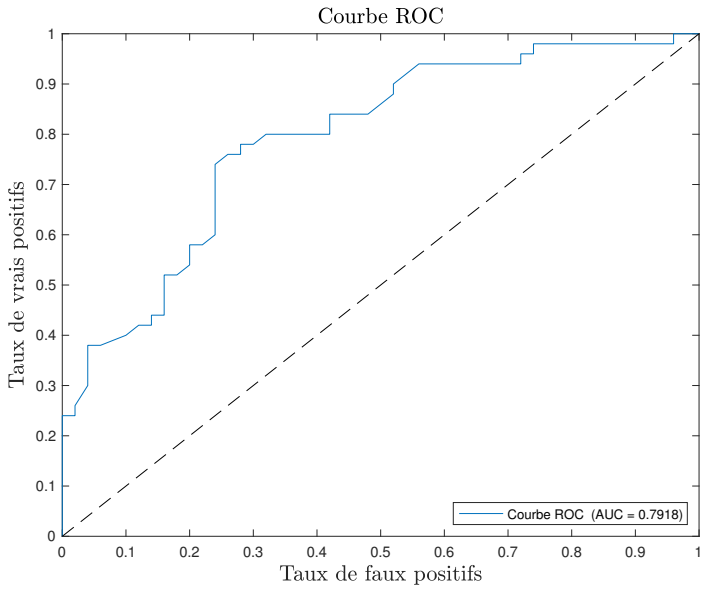


FIGURE 2.6 – Courbe ROC et son aire pour un exemple donné dans l'aide Matlab [21].

La courbe ROC peut être aisément étendue aux classifeurs multi-classes en rendant binaire les sorties du classifieur, c'est-à-dire que pour une classe considérée, cette classe correspondra à « vrai » et toutes les autres à « faux » .

2.3 Conclusion

Dans ce chapitre, nous venons de présenter les notions nécessaires à la poursuite de ce travail. En effet, nous avons introduit dans la première partie de celui-ci, l'opérateur de Koopman, ainsi que ses modes et valeurs propres. Ces quantités peuvent être extraites d'une dynamique grâce à l'algorithme de décomposition en modes dynamiques. La méthode d'identification spectrale des réseaux, quant à elle, permet de faire le lien entre le spectre lié à l'opérateur de Koopman et celui lié à un réseau dont est issu la dynamique relative aux données traitées par la méthode DMD.

Les données extraites grâce à l'algorithme DMD peuvent être utilisées avec des méthodes de classification supervisée. En effet, le but de ce travail est d'effectuer une classification spectrale des réseaux. La méthode que nous considérerons essentiellement pour la classification, les Extra-Trees, est une variante des forêts aléatoires dans laquelle l'aléatoire intervient essentiellement dans le choix des caractéristiques pour diviser les nœuds. Nous avons également présenté deux autres méthodes, les arbres de décision et la régression logistique, que nous utiliserons essentiellement dans un but de comparaison avec la première. L'introduction de méthodes de classification nous a amené à présenter deux outils permettant de mesurer la qualité de nos classifieurs, la matrice de confusion et la courbe ROC.

Dans le chapitre suivant, nous exploiterons toutes les notions présentées dans ce chapitre sur les données simulées par un modèle que nous avons introduit au chapitre précédent, ainsi que sur les données IRMf.

Chapitre 3

Résultats sur les données simulées

Dans ce chapitre, nous utilisons tous les outils présentés dans les chapitres précédents. En effet, grâce à la modélisation DCM, nous pouvons simuler des données IRMf. Nous appliquons ensuite l'algorithme de décomposition en modes dynamiques sur ces données afin d'extraire des informations sur le spectre du réseau sur lesquelles nous procéderons à une classification Extra-Trees. Tous les résultats ont été obtenus via Matlab_R2018a et les codes correspondants se trouvent en Annexe B.

3.1 Simulations

Nous avons présenté au chapitre 1.2 la manière dont peuvent être modélisées les interactions entre les neurones. Pour rappel, l'imagerie par résonance magnétique fonctionnelle mesure les variations hémodynamiques de certaines zones du cerveau, grâce au niveau d'oxygénation de ces dernières et nous fournit le signal BOLD. Nous modélisons ce signal BOLD grâce au modèle DCM. Cette modélisation nécessite de fixer un réseau, c'est-à-dire la matrice d'adjacente et les interactions entre les régions, ainsi que la fonction stimulus qui permettra la simulation de la dynamique du réseau.

Le but de ces simulations est d'avoir une situation simplifiée par rapport à nos données afin d'aiguiser les outils de classification. Dès lors, nous ne considérons que deux réseaux, simulant deux états de conscience différents. Ainsi, nous entraînerons une classification simplifiée sur deux classes, avant de l'étendre à quatre classes avec nos données.

3.1.1 Choix des paramètres

Dans la modélisation DCM, des paramètres interviennent à différents niveaux. Nous les avons fixés sur base de l'article [13]. Nous reprenons ci-dessous les paramètres communs pour toutes les simulations des réseaux. Pour ceux qui varient d'une simulation à l'autre, typiquement les valeurs de la matrice d'adjacence de l'équation d'état neuronal, ceux-ci seront fixés plus loin.

Équations d'état neuronal

Les équations d'état neuronal modélisent les interactions entre les différentes régions cérébrales, ainsi que les changements dus aux différents stimuli agissant sur le réseau. C'est grâce

à ces équations que nous pouvons définir les réseaux que nous considérerons. Notre but étant de classer deux réseaux, nous devons en créer deux distincts. Il va de soi qu'au moins ceux-ci sont différents, au plus la classification sera aisée. Il faudra donc que ceux-ci soient relativement semblables.

Nous utilisons une version simplifiée de la modélisation DCM afin de modéliser les réseaux, à savoir que nous ne considérons qu'une seule fonction stimulus en entrée,

$$u(t) = 1 + 0,5 \sin 5t.$$

Cela implique que par rapport à l'équation (1.1), nous n'aurons qu'une seule matrice B, puisque cette dernière introduit les changements de couplage dus à l'entrée u. Le choix de cette fonction stimulus vient du fait que nous voulons une fonction périodique afin de représenter les activations répétitives des zones cérébrales. Nous simplifions également nos simulations par rapport aux données réelles en ce qui concerne le nombre de régions cérébrales considérées. Nous n'en prendrons ici que dix.

Notre but étant de capturer les changements qu'implique une variation de réseau, nous considérerons deux matrices d'adjacence A différentes. En effet, prendre deux matrices A différentes va modifier le réseau sous-jacent, et nous obtiendrons deux jeux de données différents.

Pour résoudre cette équation, nous devons définir un état initial. Celui-ci est sélectionné aléatoirement de telle manière que chacune de ses composantes est comprise dans l'intervalle $]0, 1[$. Dans toutes les données obtenues par mesure, du bruit vient parasiter les captures, c'est pourquoi nous ajoutons à la dynamique donnée par ces équations un bruit de la forme

$$\eta = 0,5 \times 10^{-1} x(t) \odot \varepsilon,$$

où ε est une matrice de la même taille que x dont les éléments suivent une loi normale centrée réduite $\mathcal{N}(0, 1)$ et $\odot$ représente la multiplication élément par élément.

Modèle hémodynamique

Les équations du modèle hémodynamique possèdent cinq paramètres qui n'ont pas encore été définis. Il y en a deux pour les équations qui modélisent les réponses vasculaires et trois pour le modèle ballon. Ces valeurs sont reprises dans la Table 3.1.

Les conditions initiales des équations d'état neurovasculaire, le signal vasodilatateur initial s_0 et le flux initial f_0, sont toutes deux prises de manière aléatoire de telle sorte que chacune de leurs composantes appartiennent à l'intervalle $]0, 1[$. Les équations du modèle ballon ont également des conditions initiales v_0, le volume sanguin initial, et q_0, la quantité en désoxyhémoglobine initiale, prises aléatoirement telles que leurs composantes sont strictement comprises entre 0 et 1.

Tous les paramètres communs étant fixés, nous pouvons définir les deux réseaux que nous considérons.

Équations d'état neurovasculaire	Modèle ballon
$\kappa = 0,8$	$\alpha = 0,32$
$\gamma = 0,4$	$\tau = 0,98$
	$E_0 = 0,8$

TABLE 3.1 – Table reprenant les paramètres utilisés dans le modèle hémodynamique de la modélisation *Dynamic Causal Modelling*.

3.1.2 Premier réseau - Premier état de conscience

Dans ce premier réseau, qui correspond à notre premier état de conscience, nous voulons un réseau où les régions de neurones ne sont pas trop peu connectées entre elles, c'est pourquoi nous avons choisi un réseau complet, c'est-à-dire que tous les nœuds sont connectés les uns aux autres. Le réseau tel que nous le considérons est représenté à la FIGURE 3.1. Chaque arête possède un poids qui suit une loi uniforme $U[0,1]$. Cependant, la matrice d'adjacence doit être stable, c'est-à-dire que ses valeurs propres doivent être à partie réelle négative, c'est pourquoi le poids des boucles des nœuds ne suivent pas la même loi, mais sont tels que la stabilité du réseau est vérifiée. Cette stabilité est validée par les résultats de l'ANNEXE A.1.1. Nous y avons également placé les influences qu'impose l'entrée u, la matrice C de notre modèle, ainsi que les influences de cette entrée au travers des différents couplages entre les régions du réseau, c'est-à-dire la matrice B.

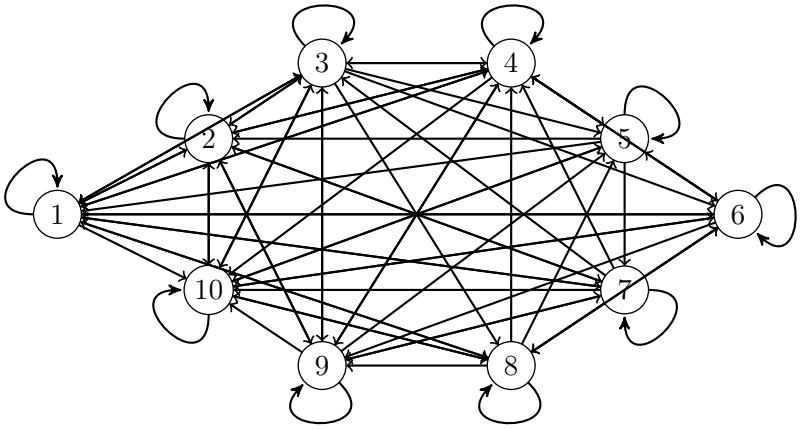


FIGURE 3.1 – Représentation schématique du premier réseau simulé. Celui-ci est relativement bien connecté afin de simuler au mieux le réseau cérébral d'un patient conscient. Les poids des arêtes suivent une loi uniforme $U[0,1]$ et ceux des boucles sont tels que la matrice d'adjacence du réseau est stable.

Nous associons ce premier réseau à l'état de conscience d'un patient dit « conscient ». En effet, un cerveau conscient possède des connexions relativement nombreuses entre ses différentes zones actives [16]. Dans la FIGURE 3.2 sont représentés les signaux BOLD issus de la modélisation

DCM pour le premier état considéré. Nous avons dix signaux, chacun d'entre eux correspondant au signal BOLD d'une des régions du réseau.

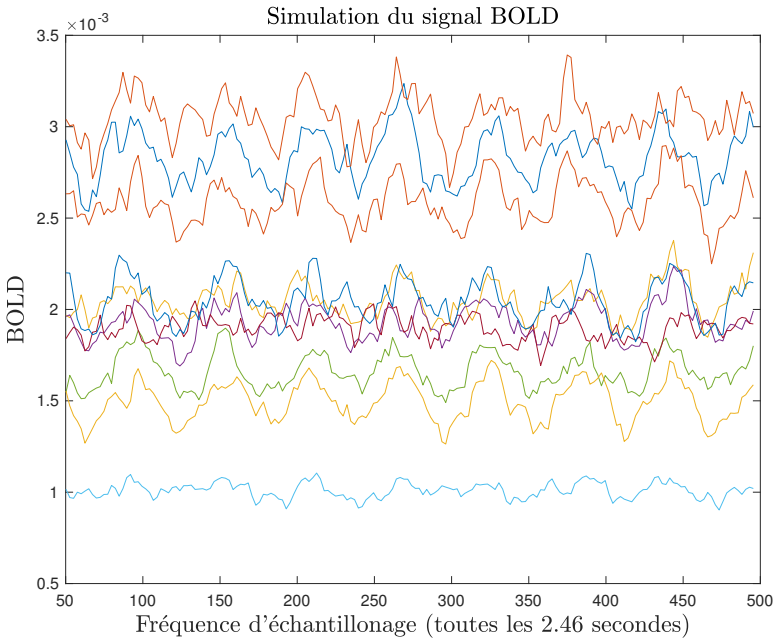


FIGURE 3.2 – Simulation du signal BOLD pour les dix régions considérées du premier réseau. Le transitoire du signal a volontairement été omis afin de ne pas observer un effet d'aplatissement des différents signaux dû à l'échelle des données.

3.1.3 Deuxième réseau - Deuxième état de conscience

Pour ce deuxième réseau, c'est-à-dire notre deuxième état de conscience, nous voulons un réseau dont les régions sont moins connectées entre-elles que pour le premier. Par conséquent, la matrice d'adjacence du réseau sera plus creuse que dans le cas précédent. La stabilité de la matrice d'adjacence du réseau est toujours assurée par le résultat de l'ANNEXE A.1.1. Les matrices B et C, nécessaires à la simulation du réseau, sont dans cette même annexe.

Tout comme le premier réseau simulé, nous pouvons associer celui-ci à un état de conscience. De par ses connexions inter-régionales moins nombreuses, nous l'associons à un état d'inconscience. Nous pouvons observer la connectivité plus légère sur la FIGURE 3.3. Comme pour le réseau précédent, les différents poids des arêtes suivent une loi uniforme $U[0, 1]$ et ceux des boucles sont tels que la stabilité du réseau est assurée.

Le modèle *Dynamic Causal Modelling* nous fournit à nouveau le signal BOLD de chacune des dix régions considérées. Ceux-ci sont représentés sur la FIGURE 3.2. Pour rappel, nous avons pris comme contrainte que les deux réseaux doivent êtres relativement différents afin de modéliser deux états distincts, mais doivent également être assez semblables afin que la classification spectrale ait un réel intérêt. Dans le cas contraire, une méthode visuelle de classification devrait sans doute faire l'affaire. Lorsque nous regardons les deux simulations, nous observons des petites

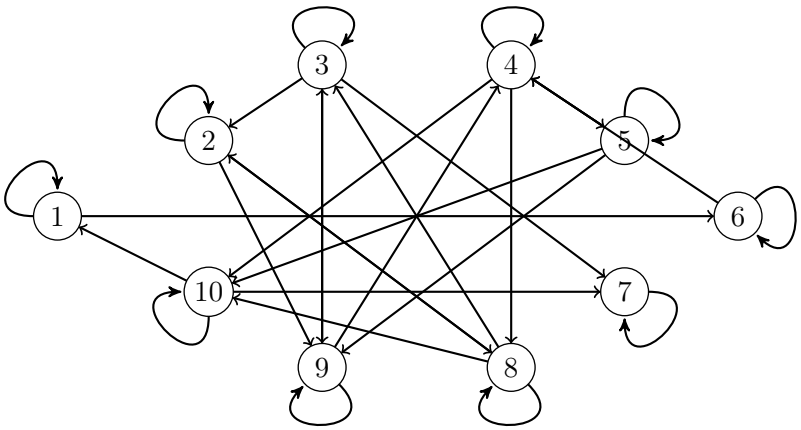


FIGURE 3.3 – Représentation schématique du second réseau simulé. Les poids des arêtes suivent une loi uniforme $U[0,1]$ et ceux des boucles sont tels que la matrice d'adjacence du réseau est stable. Les connexions entre les différents groupes de nœuds ne sont pas très nombreuses, afin de simuler au mieux l'état d'inconscience.

différences visuelles des signaux, mais à première vue, il n'est pas évident de différencier les deux dynamiques. En effet, nous avons effectué un décalage des données de $3,5 \times 10^{-3}$ vers le bas afin d'avoir des signaux BOLD plus semblables encore. Notons que ce shift n'influencera pas la classification tant que nous ne considérons pas le mode associé à la valeur propre $\lambda = 0$, car celui-ci capture la hauteur des signaux.

3.1.4 Classification

Nous avons simulé deux réseaux correspondant à deux états de conscience d'un patient. Pour pouvoir entraîner au mieux la classification, il est nécessaire de disposer de plus d'un patient. Nous avons donc effectué une simulation de vingt patients, chacun disposant des deux réseaux distincts (un pour l'état « Conscient » et un pour l'état « Inconscient »). Les différences entre ces patients sont assurées par les conditions initiales aléatoires présentées précédemment.

Choix des variables

Avant de lancer la classification proprement dite, nous devons sélectionner différentes variables que nous utiliserons avec les *Extra-Trees*. Pour ce faire, nous utilisons au préalable l'algorithme de décomposition en modes dynamiques afin d'extraire valeurs propres et modes de l'opérateur de Koopman associés à la dynamique. Nous représentons les valeurs propres et le mode dominant dans la FIGURE 3.5.

Sur la FIGURE 3.5 (b), nous pouvons observer les trois premières composantes du mode dominant, c'est-à-dire celles associées aux trois premières régions (ROI), pour tout les patients. Si nous regardons la FIGURE 3.5 (a) qui reprend toutes les composantes du mode dominant du premier patient, nous pouvons voir que les deux premières composantes du mode dominant sont plus

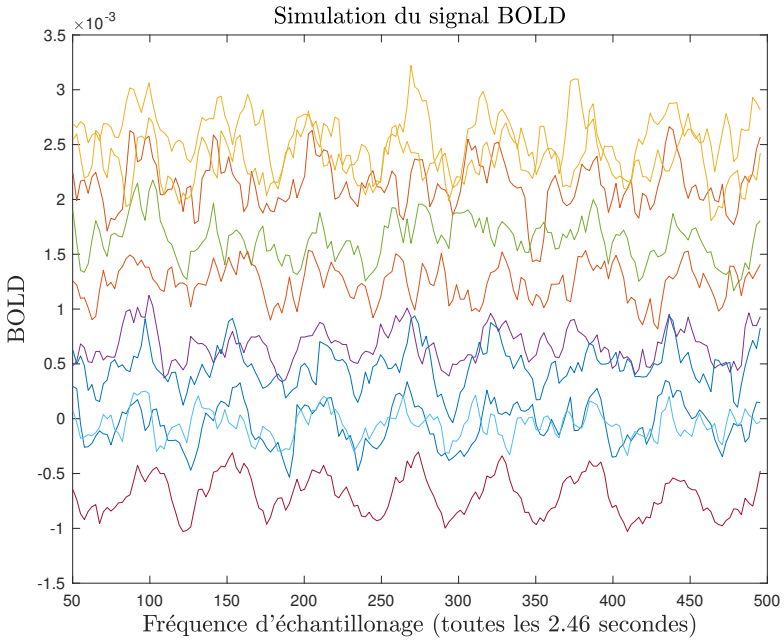


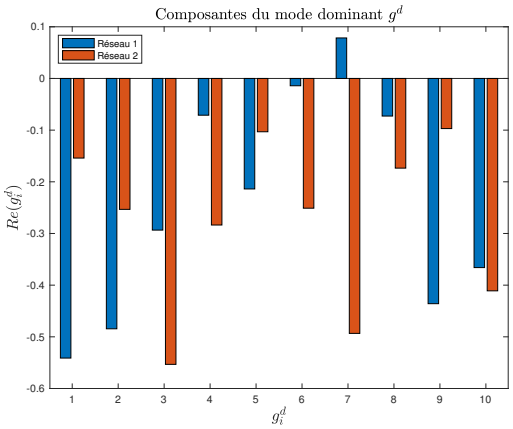
FIGURE 3.4 – Simulation des signaux BOLD des dix régions considérées du premier réseau, sans les transitoires et décalés vers le bas.

négatives pour le premier réseau que pour le deuxième. Nous retrouvons bien ce phénomène sur la FIGURE 3.5 (b), tandis que la troisième composante est plus négative pour le deuxième réseau.

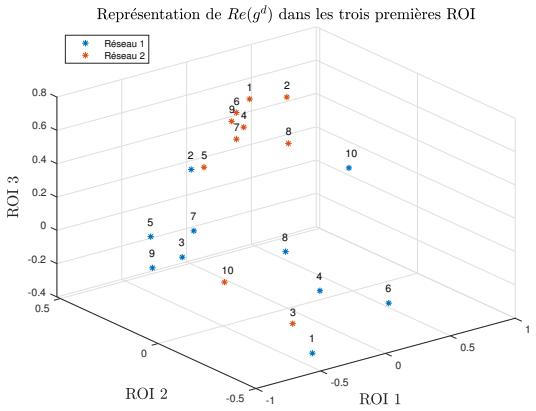
La FIGURE 3.5 (b) nous montre la répartition des modes dominants de chaque patient, dans l'espace composé par les trois premières régions. Nous pouvons observer un clustering entre les deux réseaux considérés. En effet, les points bleus, les patients conscients (le premier réseau), et les points rouges, les patients en état végétatif (le second réseau), peuvent être regroupés en deux classes distinctes sans trop de difficulté. Nous pouvons également remarquer sur la FIGURE 3.5 (c) que le spectre des deux réseaux possède des points communs, notamment la valeur propre de valeur nulle, mais également des points distincts qui vont nous intéresser pour la classification. Dès lors, effectuer une classification sur base des éléments spectraux est cohérent.

Pour la première classification, nous devons sélectionner des variables de manière arbitraire. Sur base des graphes de la FIGURE 3.5, nous décidons d'utiliser toutes les informations fournies par le mode dominant et par les valeurs propres. Ainsi, nous prenons les vingt et une variables

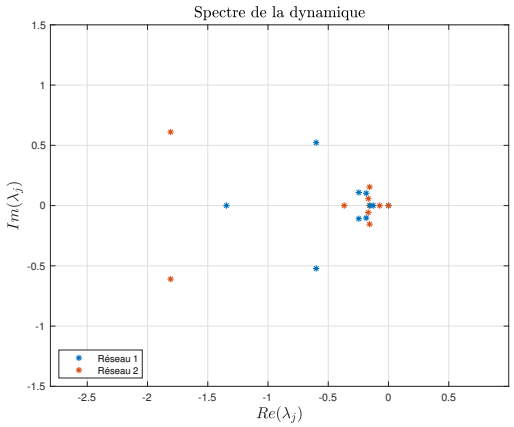
— $|\overline{\lambda}|$, c'est-à-dire le module de la moyenne des valeurs propres (en temps continu),
— $\mathfrak{Re}(g_i^d),\ i \in \{1, \ldots, 10\}$, c'est-à-dire la partie réelle de la composante du mode dominant dans chaque région,
— $\mathfrak{Im}(g_i^d),\ i \in \{1, \ldots, 10\}$, c'est-à-dire la partie imaginaire de la composante du mode dominant dans chaque région.



(a) Partie réelle du mode dominant, répartie selon ses régions.



(b) Partie réelle du mode dominant de chaque patient, selon les trois premières régions.



(c) Spectre le l'opérateur de Koopman.

FIGURE 3.5 – Sur les trois images, la couleur bleue est associée au premier réseau, et la couleur rouge au deuxième. L'image (a) représente les parties réelles de chaque composantes du mode dominant, pour le premier patient. L'image (b) représente les trois premières composantes du mode dominant de chaque simulation, c'est-à-dire de chaque patient simulé. L'image (c) représente le spectre des deux réseaux.

Rappelons que le mode dominant est le mode associé à la valeur propre dont la partie réelle est la plus grande. Nous excluons cependant le mode associé à la valeur propre $\lambda = 0$. En effet, le mode associé à cette valeur propre ne capture pas la dynamique mais la hauteur du signal, comme nous l'avons signalé dans la section 3.1.3.

Classification

Commençons par regarder la matrice de confusion de la FIGURE 3.6. Le test a une *accuracy* de 75%, ce qui est relativement bon pour une classification dichotomique. En effet, là où 50% d'*accuracy* signifie que nous avons une chance sur deux d'attribuer la bonne classe pour un pa-

tient donné, nous avons ici trois chances sur quatre de le mettre dans la bonne catégorie.

Regardons la première classe, c'est-à-dire la classe « Conscient ». La sensibilité du test pour cette première classe est de 80%, ce qui est plutôt bon, mais sa spécificité est légèrement moins bonne, étant donné les 70%. Rappelons que la sensibilité d'un test est sa capacité à prédire qu'un patient est de la classe « Conscient » (dans ce cas-ci) lorsque ce patient appartient bien à cette classe. La spécificité quant à elle est la capacité du test à déterminer qu'un patient appartient à la seconde classe, à raison. Dès lors, si nous voulons que cette première classification soit associée à l'attribution de la classe des patients conscients, elle prédira correctement cette classe à hauteur de 80%, tandis qu'elle considèrera que les patients sont dans un état d'inconscience avec raison dans 70% des cas.

La précision de la classe des patients en état de conscience est de $72,7\%$, c'est-à-dire que parmi tous les patients que la méthode a considéré dans un état de conscience, seulement $72,7\%$ d'entre eux le sont réellement. La précision concernant les patients inconscients est légèrement meilleure, avec $77,8\%$.

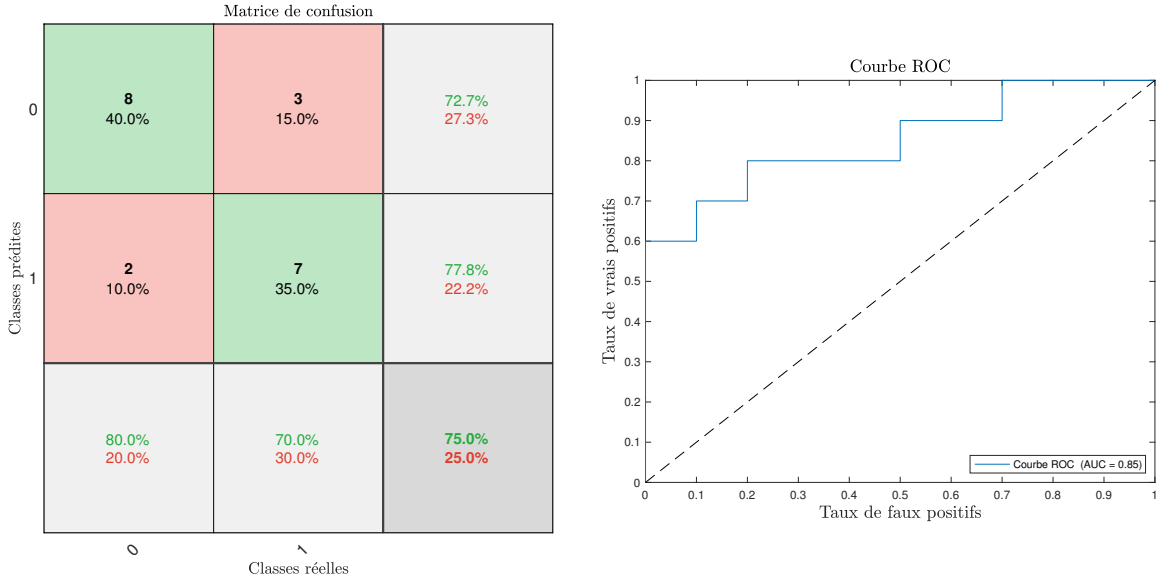


FIGURE 3.6 – Matrice de confusion (à gauche) et ROC Curve (à droite) relatives à la première classification. La classe « 0 » correspond aux patients conscients, tandis que la classe « 1 » est la classe des patients dans un état d'inconscience.

La courbe ROC de la FIGURE 3.6 nous permet de représenter la capacité du test à faire la distinction entre les deux classes. Au vu de l'allure de la courbe, nous pouvons dire que la méthode est performante. La courbe est en effet plus proche du coin supérieur gauche du graphe que de la diagonale. De plus, l'aire sous la courbe ROC est de $0,85$, qui est plus proche de 1, l'aire sous une courbe ROC d'une méthode parfaite, que de $0,5$, l'aire sous une courbe ROC d'une méthode purement aléatoire.

Ces résultats peuvent sembler satisfaisants, mais nous aimerions les améliorer. En effet, les deux états de conscience considérés sont associés à des réseaux qui peuvent sembler similaires au vu du signal BOLD, mais en réalité fort différents sur plusieurs points. Premièrement, la simulation DCM, comme nous l'avons rappelé au début de ce chapitre, a été fortement simplifiée, ce qui ne modélise pas de manière précise toutes les interactions d'un réseau complexe tel que le cerveau humain. Deuxièmement, nous n'avons simulé les données que pour dix patients, ce qui peut sembler peu pour entraîner le modèle de classification. C'est pourquoi nous allons réaliser une autre classification afin d'améliorer ces résultats.

Sur la FIGURE 3.7, nous pouvons observer la répartition de l'utilisation des variables dans la classification des deux états. Par exemple, la sixième variable, $\mathfrak{Re}(g_5^d)$, est celle qui a été la plus utilisée pour classer les deux états, tandis que la vingtième variable, $\mathfrak{Im}(g_{10}^d)$, est celle qui a été la moins sollicitée. Cette répartition peut nous aider à sélectionner des variables pour améliorer la classification. En effet, nous avons tendance à garder uniquement les variables les plus utilisées dans l'espoir qu'elles améliorent le modèle.

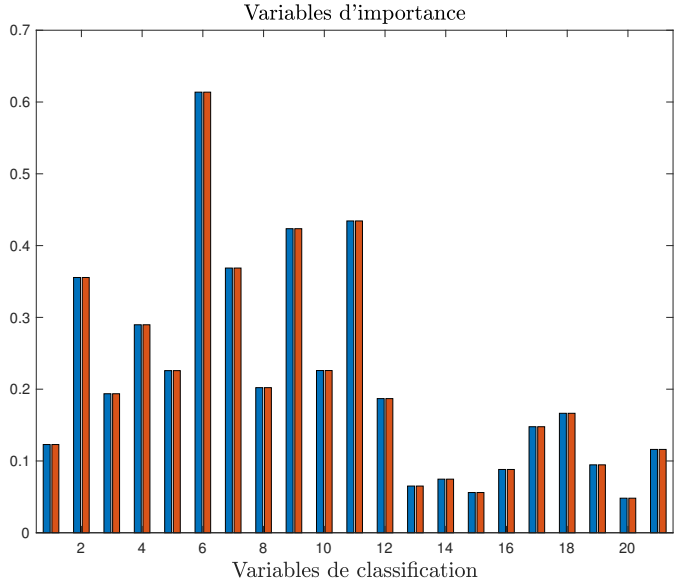


FIGURE 3.7 – Fréquence d'utilisation de chaque variable utilisée pour la première classification. Les variables sont représentées par un numéro, qui correspond à leur position dans le vecteur composé des variables de classification. Les barres bleues correspondent au premier réseau simulé, et les rouges au second.

En se basant sur les variables d'importance, nous décidons de garder celles dont la distribution est supérieure ou égale à $0,2$, c'est-à-dire

— $\mathfrak{Re}(g_i^d),\ i \in \{2, 3, 4, 5, 6, 8, 9, 10\}$,
— $\mathfrak{Im}(g_1^d)$.

Les résultats sont exactement les mêmes que pour le choix de variables précédent, à l'exception de l'aire sous la courbe ROC qui a augmenté de $0,04$ unités. Nous avons pour la première classe toujours huit patients correctement classés contre trois classés de manière incorrecte, et pour la deuxième classe, nous en avons toujours sept contre trois. Les résultats de cette classification ceux des suivantes sont disponibles dans l'Annexe A.2.

Nous allons tenter d'améliorer les résultats en sélectionnant d'autres variables. Dans ce cas-ci, nous ne prenons plus les parties réelles et imaginaires du mode dominant comme dans le premier cas, mais les modules et arguments du mode dominant, c'est-à-dire

— $|\overline{\lambda}|$, c'est-à-dire le module de la moyenne des valeurs propres (en temps continu),
— $|g_i^d|,\ i \in \{1, \dots, 10\}$, c'est-à-dire le module de la composante du mode dominant dans chaque région,
— $\arg(g_i^d),\ i \in \{1, \dots, 10\}$, c'est-à-dire l'argument de la composante du mode dominant dans chaque région.

Les résultats de cette classification ont été légèrement améliorés par rapport à celles d'avant. En effet, quelle que soit la classe considérée, nous avons huit patients associés à la bonne classe, et deux qui se sont vus attribuer la mauvaise. Dès lors, sensibilité, spécificité et exactitude sont tous de 80%. L'aire sous la courbe ROC a cependant légèrement diminué, à savoir qu'elle est de $0,82$ contre $0,89$ à la classification précédente. Nous pouvons conclure que cette classification est meilleure du point de vue de la matrice de confusion, c'est-à-dire que concrètement, les patients ont été mieux classés qu'auparavant, mais qu'elle est moins performante d'un point de vue courbe ROC, c'est-à-dire que indépendamment de la population de chaque classe, la classification est moins bonne que les précédentes.

Dans le but d'améliorer le modèle, nous sélectionnons dans l'ensemble des variables utilisées ci-dessus celles dont la distribution est supérieure à $0,2$, c'est-à-dire

— $|g_i^d|,\ i \in \{3, 5, 6, 9\}$,
— $\arg(g_i^d),\ i \in \{1, 5, 6, 8\}$.

Grâce à la Figure 3.8, nous pouvons affirmer que nos résultats sont meilleurs. En effet, la matrice de confusion nous indique une *accuracy* de 90%, ainsi qu'une spécificité et une sensibilité de respectivement 80% et 100% pour la classe des patients conscients. La courbe ROC a également été améliorée : l'allure de la courbe est plus proche d'un classifieur parfait ; nous avons une aire sous la courbe de $0,96$.

Dans la but d'améliorer à nouveau les résultats, nous avons couplé les variables les plus utilisées parmi les quatre classifications, c'est-à-dire les variables de la deuxième et de la quatrième classification. Cependant, les résultats sont exactement les mêmes que pour la méthode précédente. Cela s'explique certainement par le fait que les variables de la deuxième classification ne sont dans ce cas-ci presque pas utilisées. Dès lors, nous pouvons essayer d'entraîner un modèle

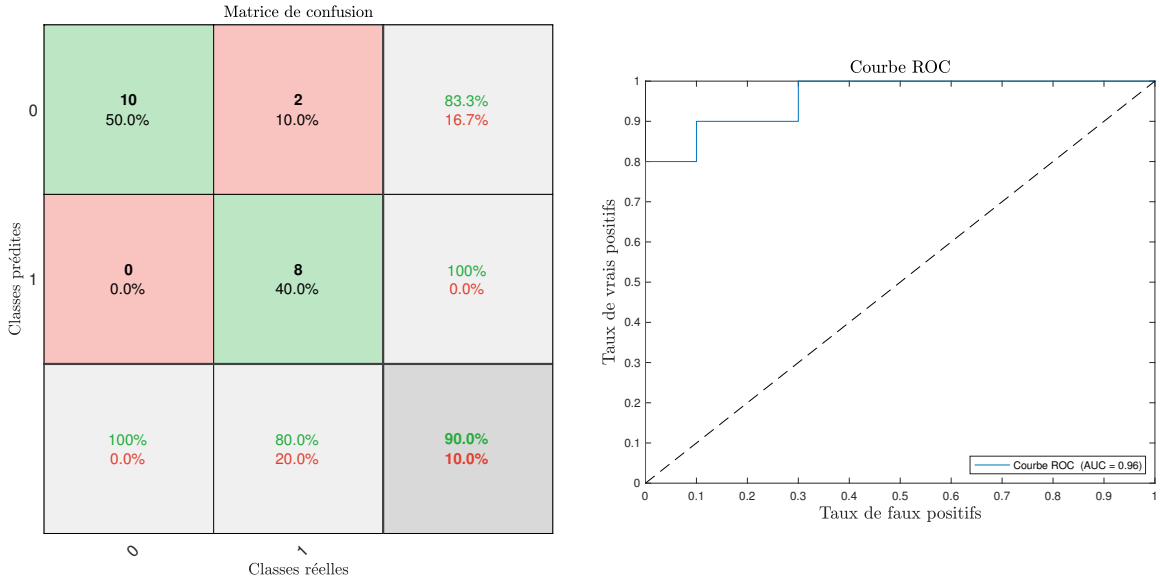


FIGURE 3.8 – Matrice de confusion (à gauche) et courbes ROC (à droite) relatives à la dernière classification retenue des simulations..

pour nos données IRMf sur base des variables utilisées pour cette dernière classification.

3.2 Conclusion

Dans ce chapitre, nous avons tout d'abord fixé les différents paramètres de la modélisation DCM, afin de simuler un jeu de données composé de dix patients, chacun d'eux ayant deux états de conscience différents.

Nous avons ensuite sélectionné différents jeux de variables afin d'entraîner le modèle des *Extra-Trees*. Ces variables ont été obtenues grâce à l'algorithme de décomposition en modes dynamiques. Par après, nous avons comparé les résultats des différentes classifications dans le but de déterminer le meilleur jeu de variables, c'est-à-dire celui qui donne la meilleure classification pour les deux classes. Ce dernier est composé du module et de l'argument du mode dominant dans certaines régions.

Dans le chapitre suivant, nous appliquerons notre méthode sur les données IRMf réelles. Ainsi, nous effectuerons sur ces données la classification *Extra-Trees* avec le jeu de variables sélectionné grâce aux simulations.

Chapitre 4

Résultats sur les données réelles

Dans ce chapitre, nous procédons à la classification sur les données réelles. Pour ce faire, nous utilisons les outils présentés précédemment, ainsi que les variables de classification retenues des simulations du chapitre précédent. Nous tentons à nouveau de prédire au mieux l'état de conscience d'un patient considéré. Nous terminons ce chapitre par une brève comparaison de la méthode *Extra-Trees* avec deux autres méthodes de classification : les arbres de décision et la régression logistique. Tous les résultats ont été obtenus via Matlab_R2018 et les codes correspondants se trouvent dans l'Annexe B.

Dans le chapitre précédent, nous avons sélectionné un jeu de variables dans le but de l'utiliser sur les données réelles. Avant de l'appliquer aux données IRMf, nous commençons par les décrire. Nous discuterons ensuite différents cas.

4.1 Données IRMf

Les médecins savent qu'il existe un vaste réseau dans le cerveau qui est constitué de régions cérébrales actives même lorsque celui-ci est au repos. Ce réseau est appelé *Resting State Network*. Ils ont d'ailleurs montré que ce réseau peut être subdivisé en cinq sous-réseaux,

— Le réseau auditif,
— Le réseau moteur,
— Le réseau visuel,
— Le réseau d'attention,
— Le réseau *Default Mode Network* (DMN).

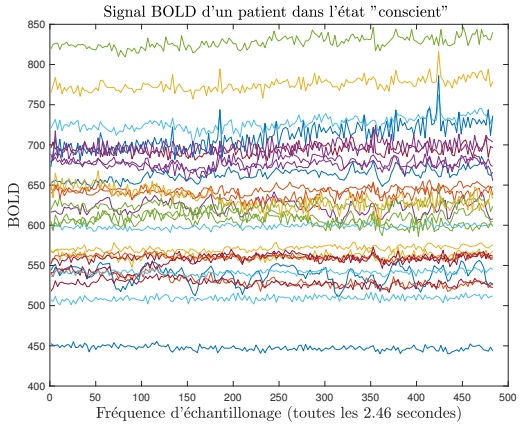
Les données dont nous disposons comportent vingt-huit régions de ce réseau DMN. C'est notamment parce que ce réseau montre des signes d'activité dans différents états de conscience que les données en sont issues. Pour rappel, ces données ont été relevées par l'équipe du Docteur Steven Laureys du groupe COMA de l'Université de Liège. Celles-ci nous ont été fournies sous trois formes, chacune correspondant à un pré-traitement différent :

— *reduced*,

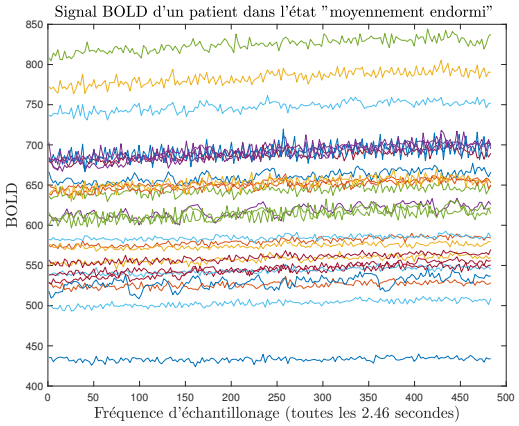
— *reduced GS*,

— *reduced GS filtered*.

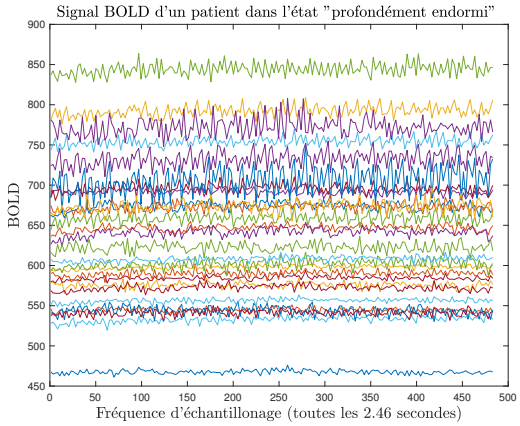
Les premières données, dénommées *reduced*, correspondent aux données de base. Les signaux BOLD des vingt-huit régions sont représentés à la FIGURE 4.1. A première vue, nous ne pouvons pas pointer de différence notable entre les quatre états de conscience. En effet, les échelles sont les mêmes pour les quatre graphes, et les allures semblables, à l'exception de légères variations par-ci par là.



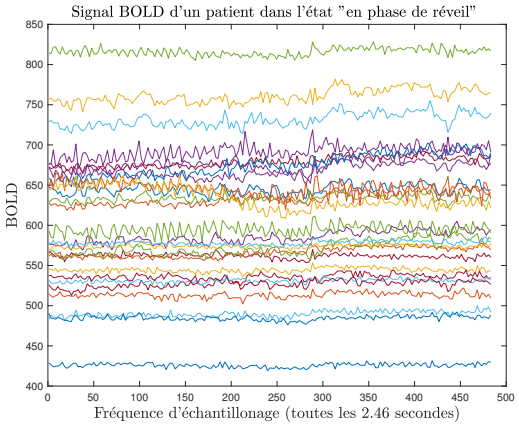
(a) « Conscient ».



(b) « Moyennement endormi ».



(c) « Profondément endormi ».



(d) « Phase de réveil ».

FIGURE 4.1 – Représentation des signaux BOLD de chacun des états de conscience du premier patient des données IRMf. Les signaux sont ceux correspondant au pré-traitement *reduced*. Chaque courbe correspond à une des 28 régions du DNM.

Les données dites *reduced GS* sont représentées à la FIGURE 4.2. Pour ces données, le pré-processing appliqué consiste à enlever le signal global, c'est-à-dire les variations globales du signal

BOLD. Nous pouvons observer que les signaux BOLD sont différents par rapport au jeu de données précédent. En effet, ceux-ci sont désormais centrés en 0.

Le dernier jeu de données, *reduced GS filterd*, s'est vu appliquer un pré-processing supplémentaire. Toutes les fréquences en dehors d'un certain intervalle ont été enlevées. En effet, ces fréquences sont considérées comme du bruit (le battement du cœur, par exemple). Ces données sont reprises sur la FIGURE 4.3.

Pour le premier jeu de données, les mesures de dix-sept patients ont été récoltées, tandis que nous en avons dix-huit pour les deux autres. Les patients qui ont participé à la récolte des données se sont vus administrer un sédatif afin de simuler l'état végétatif. Pour chaque patient, nous avons les données dans quatre états de conscience différents :

— « Conscient » : Les données ont été récoltées sur le patient lorsque celui-ci était conscient, c'est-à-dire avant l'injection du sédatif ;
— « Moyennement endormi » : Les données ont été récoltées après l'injection du sédatif sur le patient, dans le but de simuler un état de coma léger ;
— « Profondément endormi » : Les données ont été récoltées lorsque le sédatif injecté était à son efficacité maximale, dans le but de simuler l'état de coma ;
— « Phase de réveil » : Les données ont été récoltées lorsque l'effet du sédatif a commencé à s'estomper, afin de simuler la phase de réveil post coma.

Les êtres humains sont constitués différemment les uns des autres, c'est pourquoi le sédatif n'agira pas à la même vitesse pour tous les patients malgré la rapidité à laquelle son effet apparaît. Par conséquent, il est difficile de relever les mesures des patients lorsqu'ils sont exactement dans le même état. Nous aurons donc des données pour les quatre états qui varient d'un patient à l'autre. Nous avons également plus de données sur les six derniers patients considérés. En effet, nous avons des mesures en 347 points temporels pour ceux-ci, contre 197 pour les autres.

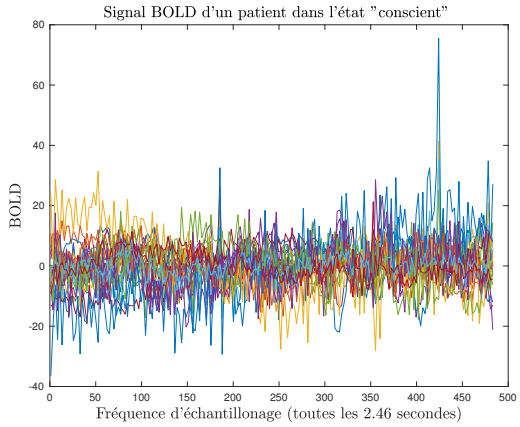
4.2 Classification

Hormis le fait que nous commençons la classification avec les mêmes variables que pour les simulations, la procédure de classification sera sensiblement similaire à celle utilisée précédemment. Ainsi, nous sélectionnons les variables spectrales sur lesquelles nous effectuons la classification Extra-Trees, puis nous analysons les différents résultats que nous avons à disposition : variables d'importance et mesures de qualité de la classification. Une petite discussion s'ensuit concernant ces résultats, et nous sélectionnons un nouveau jeu de variables si nécessaire.

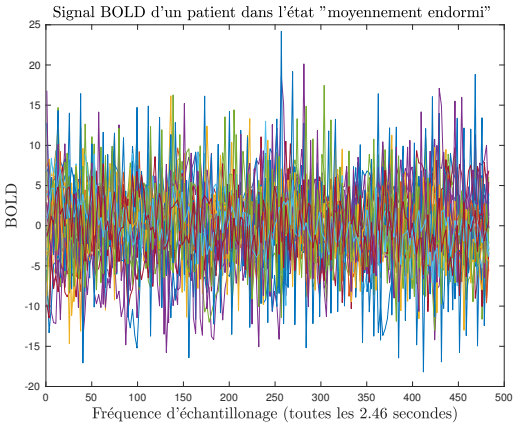
Cependant, nous ne ferons pas référence à tous les jeux de variables testés durant notre recherche et nous ne présenterons que les résultats les plus pertinents de notre recherche.

Première classification

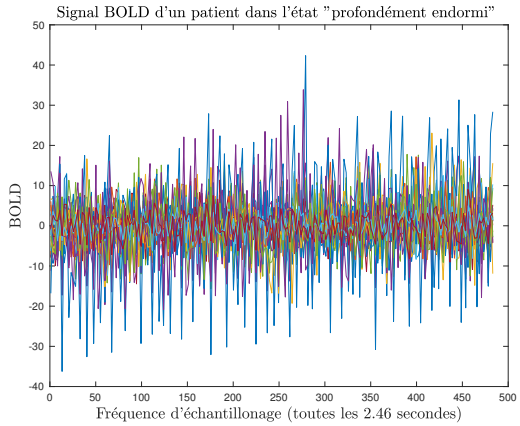
Effectuons la première classification sur les variables retenues des simulations. Pour rappel, ces variables sont les parties réelles et imaginaires du mode dominant.



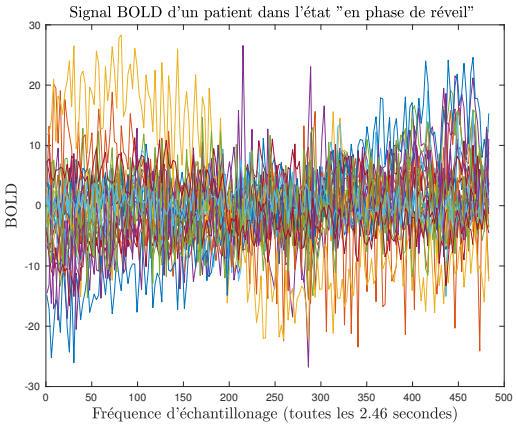
(a) « Conscient ».



(b) « Moyennement endormi ».



(c) « Profondément endormi ».



(d) « Phase de réveil ».

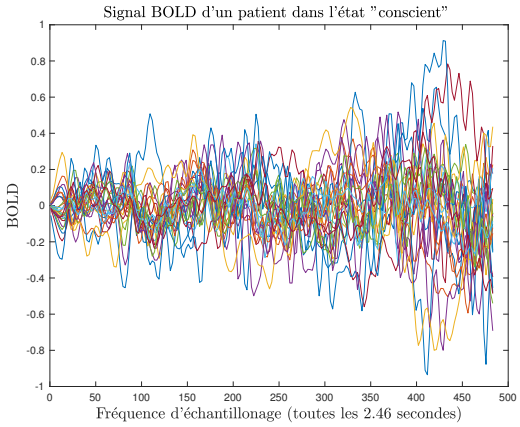
FIGURE 4.2 – Représentation des signaux BOLD de chacun des états de conscience du premier patient des données IRMf. Les signaux sont ceux correspondant au pré-traitement *reduced GS*. Chaque courbe correspond à une des 28 régions du DNM.

Lorsque nous effectuons la classification *Extra-Trees* sur base de ces variables, nous obtenons les résultats de la FIGURE 4.4. Nous y avons placé les résultats concernant chaque jeu de données. Nous allons analyser chacun d'entre-eux avant de les comparer. Notons que pour chaque jeu de données, nous avons sélectionné la meilleure classification, en retirant les variables importantes dont la fréquence d'utilisation était inférieure à $0,2$. Dès lors, pour chaque jeu de données, nous considérons des régions du cerveau différentes. Pour le jeu *reduced*, nous avons donc les variables

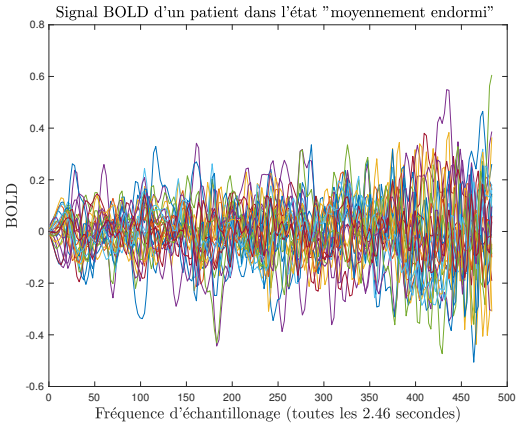
— $|g_i^d|$, $i \in \{1, \dots, 28\} \setminus \{4, 8\}$, où g_i^d est la i-ième composante mode dominant, c'est-à-dire la composante du mode dominant dans la i-ième région,

— $\arg(g_i^d)$, $i \in \{1, \dots, 28\} \setminus \{5, 6, 11, 13\}$,

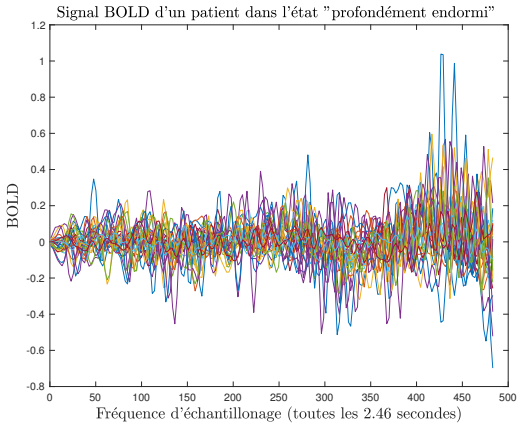
pour les données *reduced GS*,



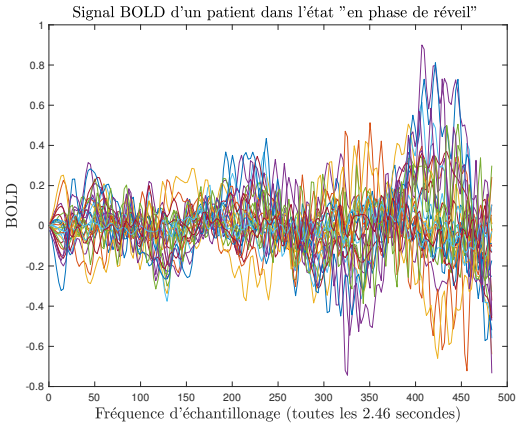
(a) « Conscient ».



(b) « Moyennement endormi ».



(c) « Profondément endormi ».



(d) « Phase de réveil ».

FIGURE 4.3 – Représentation des signaux BOLD de chacun des états de conscience du premier patient des données IRMf. Les signaux sont ceux correspondant au pré-traitement *reduced GS Filtered*. Chaque courbe correspond à une des 28 régions du DNM.

— $|g_i^d|$, $i \in \{1, \dots, 28\} \setminus \{12\}$,
— $\arg(g_i^d)$, $i \in \{1, \dots, 28\} \setminus \{3, 14, 20, 21, 22, 25\}$

et pour les données *reduced GS filtered*,

— $|g_i^d|$, $i \in \{1, \dots, 28\} \setminus \{2, 25, 26\}$,
— $\arg(g_i^d)$, $i \in \{1, \dots, 28\} \setminus \{8, 14, 25, 26\}$.

Dans cette analyse, nous procédons en deux temps. Nous commençons par analyser les résultats de la matrice de confusion en tant que tels. Nous considérons ensuite les quatre classes de manière isolée, afin d'utiliser au mieux les termes introduits dans le chapitre 2.2.4. En effet,

la spécificité d'un test telle que nous l'avons décrite nécessite une petite adaptation pour correspondre à la spécificité d'un classifieur multi-classes. Nous parlons dans ce cas de spécificité du test pour une classe donnée, et nous transformons la matrice de confusion en table de confusion.

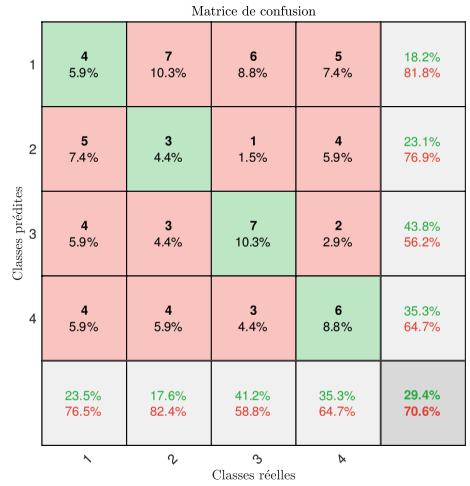
Considérons la matrice de confusion des données *Reduced* de la FIGURE 4.4. Avant de nous lancer dans une analyse plus fine des résultats, nous pouvons dire que le classifieur ne produit pas de bons résultats. En effet, l'*accuracy* du test, c'est-à-dire le taux de bonnes prédictions, n'est que de $29,4\%$.

Regardons la première ligne, la première classe prédite, c'est-à-dire les patients conscients. Les prédictions sont très mauvaises. En effet, sur les vingt-deux patients que la méthode y a classés, il n'y en a que quatre qui sont réellement conscients. En regardant les prédictions des classes suivantes, elles s'améliorent un peu. En effet, sur les prédictions de la classe « Moyennement endormi », nous avons un total de trois patients parmi treize attribués à la bonne classe. Nous avons ensuite $43,8\%$ de précision, c'est-à-dire le taux de vrais positifs, pour la classe « Profondément endormi » et $35,3\%$ pour la classe « Phase de réveil ».

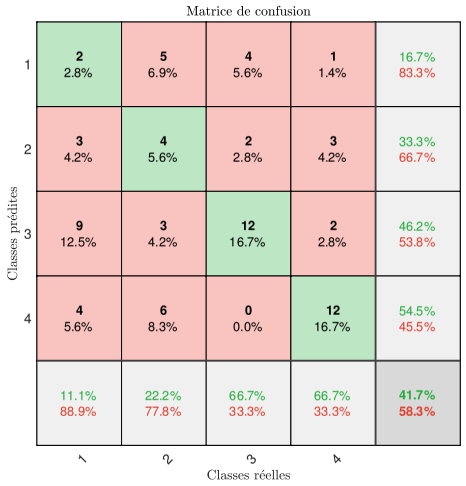
Si nous regardons les colonnes de la matrice, c'est-à-dire les classes réelles, nous pouvons dire que la méthode de classification considérée, c'est-à-dire les *Extra-Trees* couplés aux variables issues de la classification des simulations, confond un peu toutes les classes entre-elles. En effet, pour les deux premières classes, la méthode aura tendance à attribuer à un patient une classe d'une manière se rapprochant de l'aléatoire ; les patients de ces classes sont attribués de manière semblable à l'une ou l'autre classe. La dernière classe semble quant à elle ne pas trop confondre sa classe avec les patients profondément endormis, puisqu'elle n'y a placé que deux patients. Cependant, sur les dix-sept patients que comporte cette classe, il n'y en a que six de correctement placés. C'est la troisième classe qui tire le mieux son épingle du jeu. En effet, la méthode n'a l'air de confondre cette classe qu'avec la première, puisque six patient sont classés dans la catégorie « Conscient », sept dans « Profondément endormi », et seulement un dans la classe des patients moyennement endormis et trois dans ceux en phase de réveil.

Si nous considérons la courbe ROC associée à cette même classification, nous pouvons voir que les performances de la méthode ne sont pas excellentes, étant donné que les courbes de chaque classe sont assez proches de la diagonale. Si nous prenons la courbe de la classe « Profondément endormi », nous pouvons voir que ses prédictions sont meilleures que pour les autres classes. En effet, non seulement la courbe se rapproche un peu plus du coin supérieur gauche, mais son AUC, c'est-à-dire l'aire sous la courbe, est par conséquent plus élevée que pour les autres classes. Le classifieur, pour les classes « Conscient » et « Moyennement endormi » est vraiment mauvais, étant donné que l'aire sous les courbes des deux classes est de $0,47$, ce qui est en deçà des $0,5$. Le classifieur pour la quatrième classe est quant à lui légèrement meilleur qu'une méthode aléatoire puisque l'aire sous sa courbe ROC, avec une valeur de $0,56$, est à peine plus élevée que celle d'une méthode aléatoire.

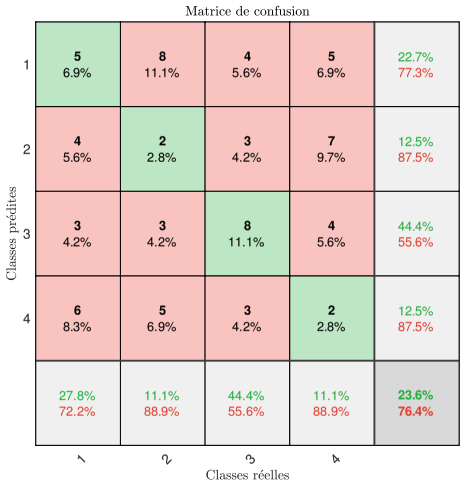
Au vu de ces résultats, affiner l'analyse de la qualité des classes produites par le modèle n'est pas nécessaire. Nous concluons que cette classification est mauvaise. Regardons à présent les résultats de la même classification, mais cette fois sur les données ayant subi le pré-traitement *Reduced GS*.



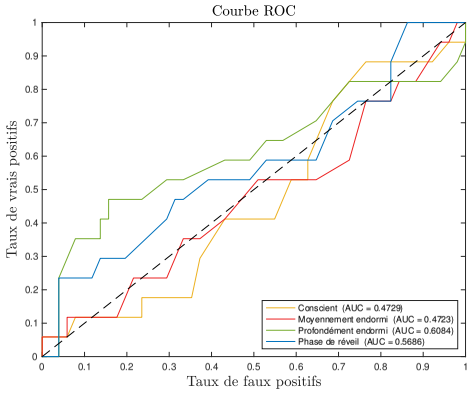
(a) Données *Reduced*.



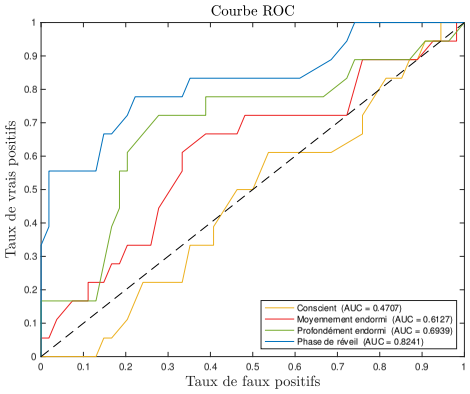
(b) Données *Reduced GS*.



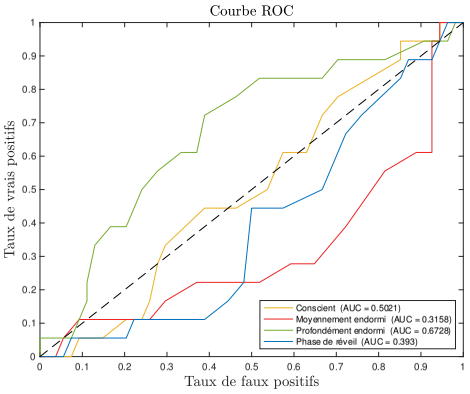
(c) Données *Reduced GS Filtered*.



(d) Données *Reduced*.



(e) Données *Reduced GS*.



(f) Données *Reduced GS Filtered*.

FIGURE 4.4 – Matrices de confusion et courbe ROC associées aux trois jeux de données disponibles. Sur les matrices de confusion, les quatre classes correspondent respectivement aux classes « Conscient », « Moyennement endormi », « Profondément endormi » et « Phase de réveil ».

Les résultats de la matrice de confusion des données *Reduced GS* semblent, à première vue, bien meilleurs. En effet, nous avons obtenu une *accuracy* de $41,7\%$. Si nous regardons les classes prédites, c'est-à-dire les lignes de la matrice, les résultats sont plus élevés que précédemment, sauf pour les patients conscients, où la précision a légèrement diminué pour atteindre $16,7\%$. La précision des trois autres classes a augmenté, en passant respectivement de $21,1\%$ à $33,3\%$ pour la deuxième classe, de $43,8\%$ à $46,2\%$ pour la troisième, et de $35,3\%$ à $54,5\%$ pour la quatrième et dernière classe. Si nous regardons les colonnes de la matrice, nous pouvons remarquer que la sensibilité de chaque classe a également été améliorée à l'exception, encore une fois, de la première classe. La sensibilité globale étant la moyenne des sensibilités, notre classification a une sensibilité

$$Se = \frac{11,1 + 22,2 + 66,7 + 66,7}{4} = 41,67\%.$$

L'*accuracy* du classifieur étant meilleure que lors de la classification précédente, nous allons regarder la matrice de confusion de manière plus fine, et se rapporter à la Table 2.1. Pour ce faire, nous devons considérer les classes une à une et regrouper les trois autres sous une seule et même classe « ¬ classe considérée ». Nous commençons par considérer la classe « Conscient ». Nous obtenons la table de confusion représentée à la Table 4.1.

	Conscient	¬ Conscient
Test positif	2	10
Test négatif	16	44

Table 4.1 – Table de confusion de la classe « Conscient » pour la première classification sur les variables *reduced GS*.

Nous avons donc, pour la classe des patients conscients, deux vrais positifs, c'est-à-dire deux patients correctement classés, seize faux négatifs, c'est-à-dire seize patients dont la méthode nous dit qu'ils ne sont pas conscients à tort, dix faux positifs, c'est-à-dire dix patients considérés « Conscient » alors qu'ils ne le sont pas. Nous avons également quarante-quatre vrais négatifs, c'est-à-dire que ces patients ne sont pas étiquetés « Conscient » à juste valeur. Ce résultat est vraiment mauvais ; la méthode n'a su classer correctement que deux patients sur les dix-huit que contient la classe des patients conscients.

La Table 4.1 nous permet de calculer la spécificité de cette classe, c'est-à-dire la probabilité qu'un patient n'appartienne pas à cette classe à juste titre. Ainsi, la spécificité de la classe sera de

$$Sp_C = \frac{44}{44 + 10} = \frac{44}{54} = 81,48\%.$$

Concernant les autres classes, leurs tables de confusion respectives sont reprises à la Table 4.2. Les spécificités respectives sont

	Moyen	¬Moyen	Profond	¬Profond	Réveil	¬Réveil
Test positif	4	8	12	14	12	10
Test négatif	14	46	6	40	6	44

TABLE 4.2 – Table de confusion des classes « Moyennement endormi », « Profondément endormi » et « Phase de réveil » pour la première classification sur les variables *reduced GS*.

$$Sp_M = \frac{46}{54} = 85,19\%, \quad Sp_P = \frac{40}{54} = 70,07\% \quad \text{et} \quad Sp_R = \frac{44}{54} = 81,48\%.$$

Regardons à présent les courbes ROC. La tendance est à nouveau la même par rapport à la classification précédente ; les AUC, à l'exception de celle de la classe « Conscient », sont respectivement plus grandes qu'avant. L'aire sous la courbe ROC de la classe « Conscient », bien que proche de $0,5$, est toujours inférieure, c'est-à-dire que la méthode telle que nous l'avons entraînée donnera des résultats pires qu'une méthode purement aléatoire pour cette classe.

Nous pouvons conclure que la méthode appliquée aux données *Reduced GS* est plus performante que cette même méthode appliquée aux données *Reduced*. Nous ne développons pas l'analyse pour la méthode appliquée aux données *Reduced GS Filtered* pour la simple et bonne raison que les courbes ROC nous donnent des résultats inférieurs à ceux de la méthode appliquée aux données *Reduced GS* et que l'*accuracy* de la matrice de confusion est également inférieur, avec $23,6\%$. Sur quatre classes, cela signifie que nous avons moins d'une chance sur quatre de prédire correctement l'état de conscience auquel appartient un patient, ce qui correspond à une méthode aléatoire.

Au vu des résultats de cette première classification, nous faisons le choix de sélectionner les données *Reduced GS* pour la suite de ce travail. Dès lors, nous développons une deuxième classification sur ce jeu de données.

Deuxième classification

Comme nous l'avons précisé au début de cette section, nous procéderons de manière légèrement différente par rapport aux classifications des simulations quant à la présentation des résultats. En effet, au cours de notre recherche, nous avons été amenés à explorer différents jeux de variables. Cependant, pour des raisons de clarté du manuscrit, nous ne présentons ici que les jeux de variables qui nous semblent les plus pertinents et qui sont associés aux classifications les plus performantes. Nous notons que les graphes d'utilisation des variables dans la classification ne sont plus utilisés explicitement, puisque comme nous venons de le signaler, les classifications qui seront désormais décrites correspondent au résultat final et cette information a donc déjà été utilisée afin d'arriver au résultat.

Nous procéderons comme pour la classification précédente en ce qui concerne l'analyse de la méthode. Cependant, il nous faut tout d'abord préciser le jeu de variables sur lequel la méthode

a été utilisée.

— $\mathfrak{Re}\,(\overline{\mu})$: la partie réelle des valeurs propres (en temps discret),
— $|g_i^d|,\ i \in \{1,\ 10,\ 14,\ 22,\ 26\}$: le module de la i-ème composante du mode dominant,
— $\arg(g_i^d),\ i \in \{1,\ 5,\ 15,\ 20\}$: l'argument de la i-ème composante du mode dominant,
— $||\tilde{g}^d||$: la norme du mode dominant rescaled,
— $\overline{\{\mathfrak{Re}(\mu)\}^2}$: la moyenne du carré de la partie réelle des valeurs propres,
— $||\tilde{g}^d||^2 - ||\tilde{g}^{d-1}||^2 - ||\tilde{g}^{d-2}||^2 - ||\tilde{g}^{d-3}||^2$, où g^{d-i} désigne le mode associé à la valeur propre dont la partie réelle est la i-ème plus grande, après celle associée au mode dominant.

Toutes ces variables sont nouvelles par rapport à la classification précédente. En effet, concernant le module et l'argument du mode dominant, nous ne tenons pas compte des mêmes régions.

Nous avons décidé d'ajouter la première variable, $\mathfrak{Re}\,(\overline{\mu})$, pour introduire les informations contenues dans les valeurs propres. Nous prenons en effet le centre de masse des valeurs propres, prises en temps discret, puisqu'une valeur propre seule n'apporte qu'une information insignifiante sur le réseau. Le centre de masse nous amène l'idée d'une fréquence moyenne du signal. L'introduction de la variable $\overline{\{\mathfrak{Re}(\mu)\}^2}$ apporte le même genre d'informations, mais en donnant davantage d'importance à la partie réelle des valeurs propres. Cette variable amène également une notion de variance, qui représente en quelque sorte la « largeur » du *cluster* de valeurs propres. La norme du mode dominant du mode rescaled a été choisie afin de tenir compte de la grandeur du mode dominant. La dernière variable, quant à elle, permet de tenir compte de la grandeur des modes les moins dominants ; nous voulions utiliser des modes associés à des valeurs propres plus éloignées de l'axe imaginaire.

Les résultats de cette deuxième classification sont repris sur la FIGURE 4.5. Nous pouvons commencer par regarder les courbes ROC. Pour les quatre classes, les aires sous les courbes sont respectivement meilleures que pour la première classification. Cette deuxième méthode de classification serait donc plus performante que la précédente.

Nous allons à présent analyser la matrice de confusion, reprise sur cette même figure. L'*accuracy* est désormais de $38,9\%$. En regardant les résultats classe par classe, nous pouvons affirmer que celle ayant la plus grande précision est, avec $52,4\%$, la classe « Profondément endormi », suivie de la classe des patients conscients avec $37,5\%$. Ce résultat est cohérent avec notre intuition que ce sont les deux classes les moins compliquées à prédire. Concernant la sensibilité de la méthode, elle vaut

$$Se = \frac{33,3 + 27,8 + 61,1 + 33,3}{4} = 38,87\%,$$

ce qui est légèrement inférieur à celle de la première classification, ce qui signifie que notre méthode actuelle est un peu moins capable de prédire correctement la classe considérée, d'un point de vue global.

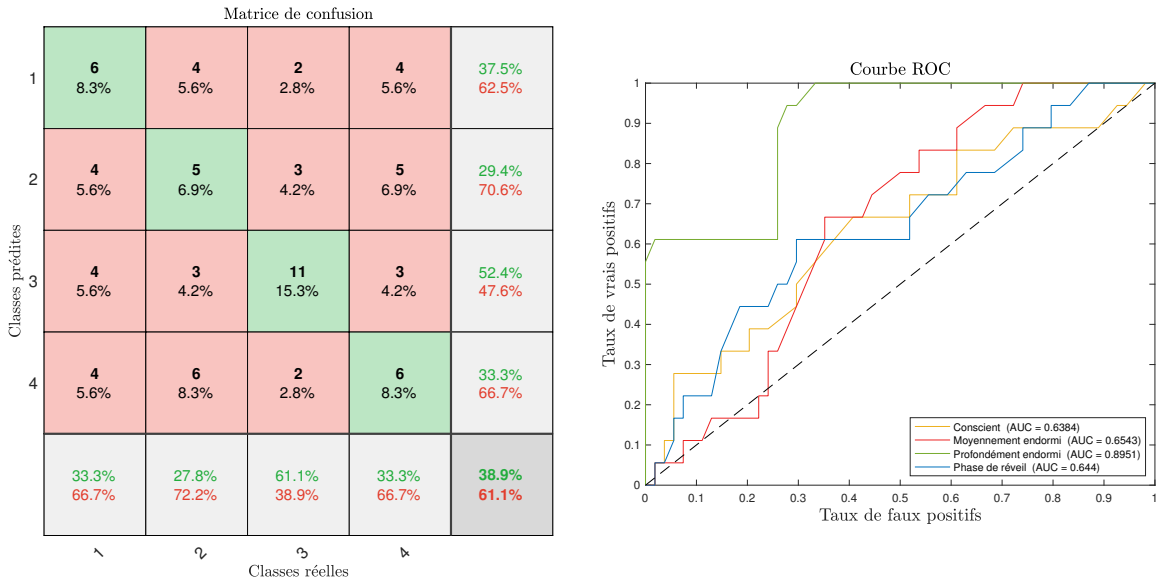


FIGURE 4.5 – Matrice de confusion et courbe ROC associées à la deuxième classification.

En ce qui concerne les sensibilités individuelles, celles des classes « Moyennement endormi » et « Profondément endormi » ont diminué par rapport à la précédente classification, ce qui signifie que lorsque la méthode prédit qu'un patient est moyennement endormi (respectivement profondément endormi), il y a un nombre moins important de patients qui le sont réellement par rapport à la première classification.

Regardons les résultats de la méthode classe par classe au travers de la table de confusion de la TABLE 4.3.

	Conscient	¬Conscient	Moyen	¬Moyen
Test positif	6	10	5	12
Test négatif	12	44	13	42

	Profond	¬Profond	Réveil	¬Réveil
Test positif	11	10	6	12
Test négatif	7	44	12	42

TABLE 4.3 – Tables de confusion pour chacune des classes de la deuxième classification.

Dès lors, nous pouvons calculer les spécificités respectives des classes,

$$Sp_C = \frac{44}{54} = 81,48\%, \quad Sp_M = \frac{42}{54} = 77,78\%, \quad Sp_P = \frac{44}{54} = 81,48\% \text{ et } \quad Sp_R = \frac{42}{54} = 77,78\%.$$

Celles-ci sont, comme pour la sensibilité, plus faibles pour les classes de patients moyennement endormis et en phase de réveil. La spécificité de la catégorie « Profondément endormi » est la même qu'auparavant, mais celle-ci est plus grande pour la classe « Conscient ». Ces constatations signifient que la méthode est moins capable de reconnaitre un négatif lorsqu'elle associe à un patient cette classe négative. Ainsi, pour la classe « Moyennement endormi » par exemple, la méthode va dans $77,78\%$ des cas classer un patient comme « ¬ Moyennement endormi » avec raison. Malheureusement, elle classera ce patient « ¬ Moyennement endormi » dans une des trois autres classes, sans pour autant le classer correctement.

Troisième classification

Le jeu de variables utilisé pour cette troisième classification est légèrement différent de celui utilisé pour la classification précédente :

- $\mathfrak{Re}\,(\overline{\mu})$: la partie réelle de la moyenne des valeurs propres,
- $|g_i^d|,\ i \in \{1,\ 10,\ 14,\ 22,\ 26\}$: le module de la i-ème composante du mode dominant,
- $\arg(g_i^d),\ i \in \{1,\ 5,\ 15,\ 20\}$: l'argument de la i-ème composante du mode dominant,
- $||\tilde{g}^d||$: la norme du mode dominant rescaled,
- $\overline{\mathfrak{Re}\left\{(\mu)^2\right\}}$: la moyenne de la partie réelle du carré des valeurs propres,
- $\overline{\mathfrak{Re}\left\{(\mu)^3\right\}}$: la moyenne de la partie réelle du cube des valeurs propres,
- $||\tilde{g}^d||^2 - ||\tilde{g}^{d-1}||^2 - ||\tilde{g}^{d-2}||^2 - ||\tilde{g}^{d-3}||^2$, où g^{d-i} désigne le mode associé à la valeur propre dont la partie réelle est la i-ème plus grande, après celle associée au mode dominant.

Les variables qui ont été introduites par rapport à la classification précédente ont été écrites en couleur bleue. Grâce à elles, nous tenons compte de la partie réelle des valeurs propres en temps discret d'une autre manière. En effet, nous tiendrons compte des valeurs propres qui ont été davantage éloignées du centre $(0, 0)$ du plan complexe, c'est-à-dire que nous tiendrons compte de leur dispersion.

Les résultats de cette classification se trouvent à la FIGURE 4.6. Nous pouvons observer sur la matrice de confusion que l'*accuracy* est meilleure que pour les méthodes précédentes, avec $45,8\%$.

Concernant la précision des classes, elle a augmenté pour les classes des patients moyennement endormis et profondément endormis, en arrivant à $38,9\%$ de précision pour la première. La classe « Profondément endormi » atteint un excellent pourcentage de précision avec $81,2\%$. La sensibilité des classes a quant à elle augmenté pour la classe « Conscient » et pour la classe « Moyennement endormi » avec $38,9\%$, ainsi que pour la classe « Profondément endormi » avec $72,2\%$. La sensibilité de la classe « Phase de réveil » a cependant diminué par rapport à la première classification, c'est-à-dire que cette méthode est moins performante quant à l'attribution

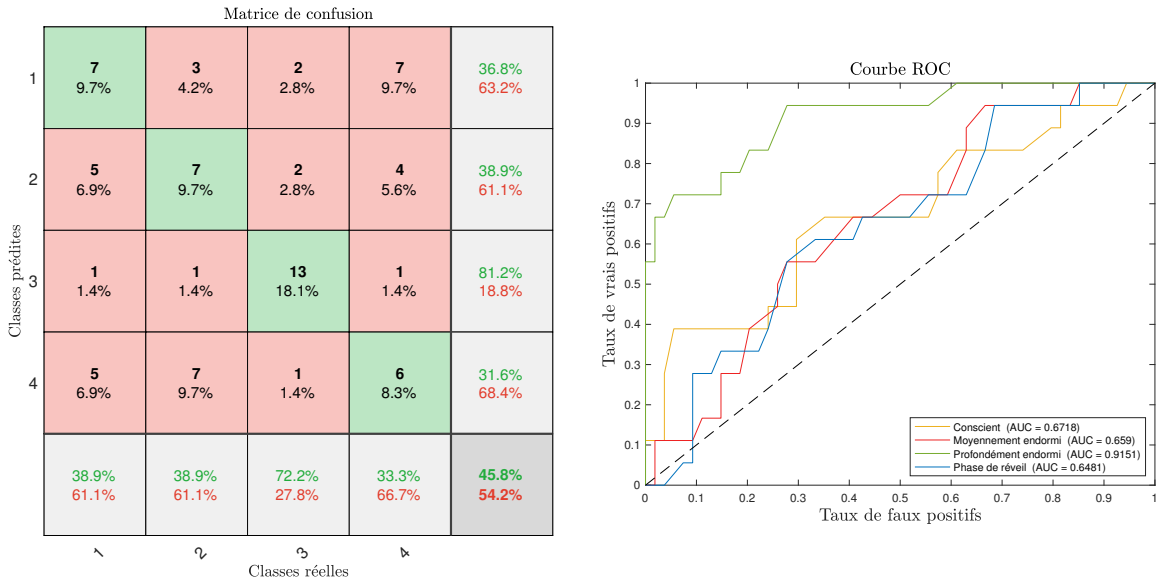


FIGURE 4.6 – Matrice de confusion et courbes ROC associées à la troisième classification.

du label « Phase de réveil » lorsque le patient est réellement dans cette classe.

La sensibilité globale du test, avec

$$Se = \frac{38,9 + 38,9 + 72,2 + 33,3}{4} = 45,82\%,$$

est également plus élevée que pour les autres classifications. Cela signifie que cette troisième méthode est globalement plus à même d'attribuer correctement la classe d'un patient que les précédentes. Nous avons une meilleure détection et une meilleure précision que pour les méthodes précédentes, à l'exception de la quatrième classe. Nous reprenons dans la TABLE 4.4 les éléments nécessaires au calcul de la spécificité de chaque classe.

Dès lors, nous avons les spécificités

$$Sp_C = \frac{42}{54} = 77,78\%, \quad Sp_M = \frac{43}{54} = 79,63\%, \quad Sp_P = \frac{51}{54} = 94,44\% \quad \text{et} \quad Sp_R = \frac{41}{53} = 77,36\%.$$

La spécificité de la classe « Profondément endormi », c'est-à-dire la capacité de la classification à ne pas classer à tort les patients qui ne sont pas endormis profondément dans cette classe, est bien meilleure que lors des précédentes classifications. La spécificité de la classe « Moyennement endormi » est quant à elle la même que lors de la première classification.

	Conscient	¬Conscient	Moyen	¬Moyen
Test positif	7	12	7	11
Test négatif	11	42	11	43

	Profond	¬Profond	Réveil	¬Réveil
Test positif	13	3	6	12
Test négatif	5	51	13	41

TABLE 4.4 – Tables de confusion pour chacune des classes de la troisième classification.

Nous pouvons à présent regarder les courbes ROC de chacune des classes. Les aires sous chaque courbe sont meilleures pour cette méthode. Ainsi, nous pouvons dire que la troisième classification est plus performante que les deux précédentes.

Comme nous avons pu le voir grâce à la matrice de confusion, la méthode est très bonne pour la classe « Profondément endormi ». Celle-ci possède d'ailleurs une AUC de $0,91$, ce qui est proche d'un classifieur parfait et son AUC de 1. Dans la matrice de confusion, nous avions une bonne précision pour cette classe, avec $81,2\%$, ainsi que de bonnes sensibilité et spécificité, avec respectivement $72,2\%$ et $94,4\%$.

De par ces résultats, cette méthode est meilleure que les autres, à l'exception de la prédiction de la classe « Phase de réveil ». Nous allons essayer d'améliorer ces prédictions, surtout pour la dernière classe, grâce à la sélection d'un nouveau jeu de variables.

Quatrième classification

Les variables utilisées pour cette nouvelle classification sont données ci-dessous.

- $|\overline{\lambda}|$: le module de la moyenne des valeurs propres,
- $\overline{|\lambda|}$: la moyenne des modules des valeurs propres,
- $|g_i^d|,\ i \in \{1,\ 10,\ 14,\ 22,\ 26\}$: le module de la i-ème composante du mode dominant,
- $\arg(g_i^d),\ i \in \{1,\ 5,\ 15,\ 20\}$: l'argument de la i-ème composante du mode dominant,
- $||\tilde{g}^d||$: la norme du mode dominant rescaled,
- $\overline{|\lambda|^2}$: la moyenne du carré des modules des valeurs propres,
- $||\tilde{g}^d||^2 - ||\tilde{g}^{d-1}||^2 - ||\tilde{g}^{d-2}||^2 - ||\tilde{g}^{d-3}||^2$, où g^{d-i} désigne le mode associé à la valeur propre dont la partie réelle est la i-ème plus grande, après celle associée au mode dominant ,
- $\mathfrak{Im}(g_{25}^d)$: la partie imaginaire de la vingt-cinquième composante du mode dominant,
- $\overline{\mathfrak{Im}(\lambda)^2}$: la moyenne du carré de la partie imaginaire des valeurs propres,
- $\overline{\mathfrak{Re}(\lambda)^2}$: la moyenne du carré de la partie réelle des valeurs propres.

Nous avons à nouveau écrit en bleu les nouvelles variables introduites dans cette classification. Dans cette nouvelle sélection, nous en avons ajouté quatre qui tiennent compte des valeurs propres en temps continu. La première, $|\overline{\lambda}|$, tient compte de la taille du centre de masse des valeurs propres, tandis que la deuxième, $\overline{|\lambda|}$, utilisera la valeur moyenne de la dispersion des valeurs propres sur la partie positive de l'axe réel. Les deux dernières vont quant à elles tenir compte des centres de masse des valeurs propres en donnant de l'importance à l'axe imaginaire et à l'axe réel. Nous avons également introduit la $25^{\text{ème}}$ région dans la classification au travers de la variable $\mathfrak{Im}(g^d_{25})$.

Sur la FIGURE 4.7, nous avons les résultats de la quatrième classification. Nous pouvons observer que la sensibilité des classes « Conscient » et « Phase de réveil » sont plus élevées que précédemment, avec respectivement $61,1\%$ et $44,4\%$. Cela signifie que la méthode est meilleure que les précédentes en ce qui concerne la bonne attribution de ces deux classes. La sensibilité de la troisième classe n'a pas changé et est toujours la meilleure avec ses $72,2\%$. En ce qui concerne la sensibilité globale, nous avons un meilleur pourcentage, avec $52,8\%$ de bonne attribution des classes aux patients, tout comme l'*accuracy*.

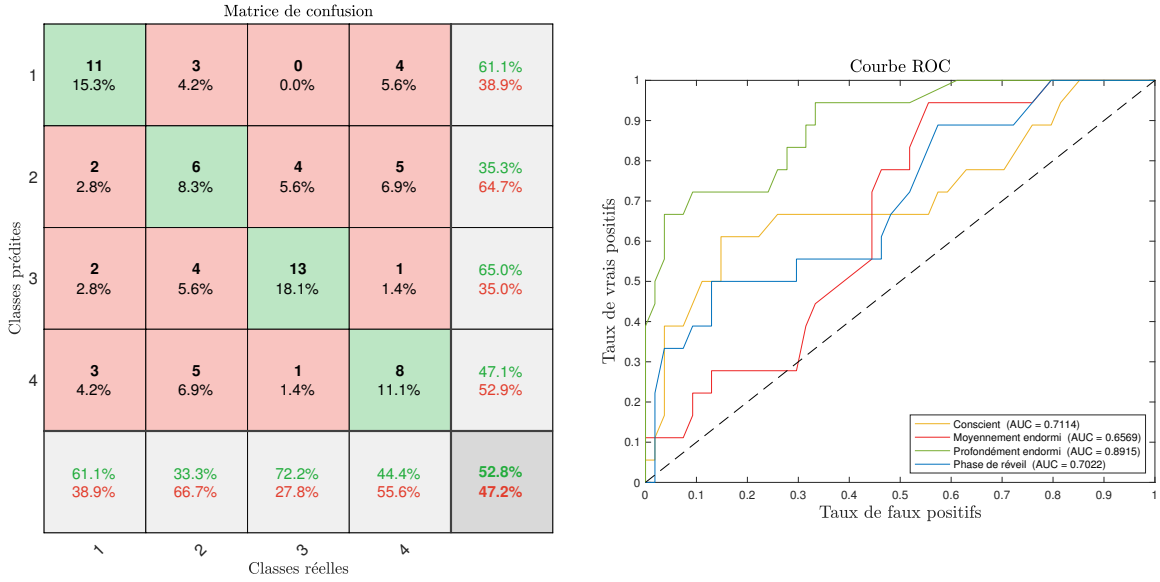


FIGURE 4.7 – Matrice de confusion et courbes ROC associées à la quatrième classification.

Les précisions des classes des patients conscients et en phase de réveil sont meilleures que pour les méthodes précédentes, avec respectivement $61,1\%$ et $47,1\%$ (qui est le même pourcentage que pour la première classification). Bien que la sensibilité de la classe « Profondément endormi » soit la même que lors de la troisième classification, sa précision a subi une certaine diminution, passant de $81,2\%$ à 65%.

Comme lors des classifications précédentes, nous reprenons dans la TABLE 4.5 les éléments nécessaires au calcul de la spécificité de chacune des classes.

	Conscient	¬Conscient	Moyen	¬Moyen
Test positif	11	7	6	11
Test négatif	7	47	12	43

	Profond	¬Profond	Réveil	¬Réveil
Test positif	13	7	8	9
Test négatif	5	47	10	45

TABLE 4.5 – Tables de confusion pour chacune des classes de la quatrième classification.

Les spécificités respectives sont

$$Sp_C = \frac{47}{54} = 87,04\%, \quad Sp_M = \frac{43}{54} = 79,63\%, \quad Sp_P = \frac{47}{54} = 87,04\% \quad \text{et} \quad Sp_R = \frac{45}{54} = 83,33\%.$$

Elles sont meilleures pour cette méthode, à l'exception de celle concernant la classe de patients dans un état de sommeil profond qui a légèrement diminué par rapport à la classification précédente.

Si nous regardons les courbes ROC, nous avons des aires qui ont augmenté uniquement pour les catégories de patients conscients et pour ceux en phase de réveil. Malgré cela, nous estimons que, d'un point de vue global, cette méthode est plus performante que les précédentes. Réalisons cependant une autre classification afin d'augmenter au mieux les bonnes prédictions sur les deux classes associées aux états de transition, c'est-à-dire les états de sommeil moyen et la phase de réveil.

Cinquième classification

Pour cette cinquième classification, nous avons un jeu de variables moins diversifié que ce que nous avons pu avoir jusqu'à présent. En effet, nous avons sélectionné

— $\overline{\mathfrak{Re}(\mu)}$: la moyenne des parties réelles des valeurs propres en temps discret,
— $|g_i^d|$, $i \in \{3, 5, 6, 7, 10, 15, 17, 18, 21, 22, 25\}$: le module de la i-ème composante du mode dominant.

Dans cette classification, il ne reste que la deuxième variable de commune à la précédente. Nous avons cependant tenu compte de plus de régions. Elles sont écrites en bleu, tout comme la nouvelle variable introduite. Dans ce jeu de variables, nous tenons à nouveau compte des valeurs

propres en temps discret au travers de la variable $\overline{\mathfrak{Re}(\mu)}$.

Lorsque nous regardons les résultats sur la FIGURE 4.8, nous pouvons nous satisfaire d'être arrivé à améliorer la classification des états de transition. En effet, comme nous pouvons le voir sur la matrice de confusion, la sensibilité de ces deux classes est passée à $55,6\%$ pour les patients moyennement endormis, et à $66,7\%$ pour ceux en phase de réveil. Les précisions ont également été améliorées, celles-ci étant respectivement de $62,5\%$ et $70,6\%$. Ces améliorations ont cependant un coût : la classification des deux autres catégories s'est détériorée (ou a stagné) par rapport à certaines classifications précédentes. La sensibilité globale de cette méthode et son *accuracy* sont néanmoins meilleures, puisqu'elles atteignent désormais les $59,7\%$.

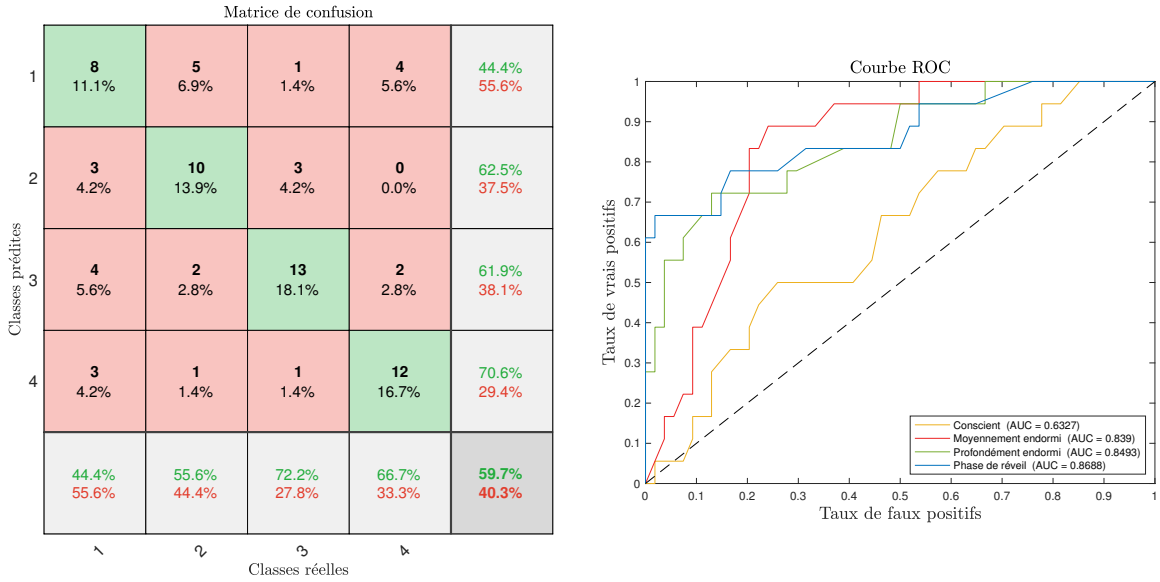


FIGURE 4.8 – Matrice de confusion et courbe ROC associées à la cinquième classification.

Les courbes ROC nous indiquent que la classification a la même performance que précédemment, sauf en ce qui concerne la classe des patients conscients qui a une aire AUC inférieure et qui est donc moins performante. De manière intuitive, nous pourrions trouver qu'il est contre-intuitif que la classification ne puisse pas mieux prédire les patients conscients. En effet, il semblerait logique que les classes de patients conscients et profondément endormis soient les plus aisées à prédire, vu leur nature extrême. Il n'en n'est cependant rien dans ce cas-ci.

Une explication de ce phénomène pourrait provenir du choix des variables utilisées pour la classification. En effet, si nous regardons les variables importantes de la FIGURE 4.9, nous pouvons voir la répartition de l'utilisation des variables dans la classification. Focalisons notre attention sur la première variable considérée, c'est-à-dire $\overline{\mathfrak{Re}(\mu)}$. Nous pouvons voir qu'elle a fortement été utilisée dans le processus de subdivision des feuilles qui déterminent la classe « Profondément endormi » mais également, dans une plus faible mesure, pour la classe « Conscient ». Si nous regardons les autres variables, la classification des patients conscients les a utilisées de manière

équivalente. Or, si la méthode avait plus utilisé certaines de ces variables dans la prédiction de cette classe de conscience, les résultats auraient peut-être été meilleurs. Nous pouvons également supposer que les variables entrées dans la méthode ne sont pas les plus adaptées pour la classification de patients conscients.

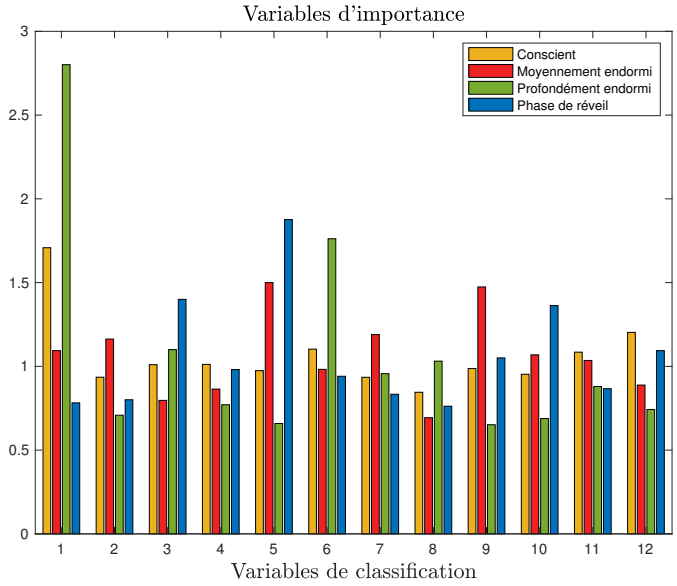


FIGURE 4.9 – Fréquences d'utilisation de chacune des douze variables utilisées pour la cinquième classification.

La difficulté de la méthode est que nous ne savons pas à l'avance quelle variable est la plus pertinente en ce qui concerne la classification de l'une ou l'autre classe. Savoir à quoi correspondent les éléments spectraux d'un point de vue cérébral pourrait probablement nous permettre de sélectionner, avec plus de pertinence, les variables que nous rentrons dans le modèle de classification. Une autre difficulté vient du fait que si nous introduisons dans un modèle deux variables qui ont été fortement utilisées dans deux classifications différentes pour une même classe, rien ne nous garanti que la méthode agira de la même façon et augmentera fortement les bonnes prédictions des classes autrefois affectées par ces deux variables.

Nous pourrions continuer à tenter d'améliorer ces classifications, mais au cours de notre recherche, nous nous sommes aperçus que les performances atteignent un plafond. Nous avons donc décidé de nous arrêter à ce stade et d'essayer de sélectionner le meilleur modèle.

4.3 Choix de la classification

Nous venons de présenter les cinq modèles de classification que nous avons préalablement sélectionnés. Comme nous venons de le voir, aucune de nos classifications ne permet d'obtenir d'excellents résultats pour toutes les classes en même temps.

De manière générale, nous avons deux types de méthodes. La première est une méthode dite de « détection ». Cette méthode a tendance à prédire plus d'éléments d'une classe que ce qu'elle ne comporte réellement. Prenons un simple exemple : la méthode cherche à prédire qu'un patient est conscient. La méthode de détection aura tendance à prédire qu'un patient est dans le coma avec un taux de faux positifs qui peut être élevé mais surtout, avec une sensibilité élevée. Le but d'une telle méthode est essentiellement de détecter si un patient est dans le coma afin de pouvoir réaliser des examens complémentaires, à l'aide d'une méthode plus précise, quant à cet état.

La deuxième méthode est justement une méthode de précision élevée. Cette méthode a pour but d'avoir un taux d'erreur minimal ; lorsqu'elle prédit qu'un patient est dans un état de conscience, ce patient l'est réellement avec une certitude assez élevée. Cependant, ce type de méthode pourrait passer à côté de patients conscients, tout simplement parce que pour eux, sa certitude n'est pas suffisante. C'est pourquoi cette méthode est dite « prudente ». Pour ce type de méthode, il vaut même mieux qu'elle omette certains patients. En effet, nous nous trouvons toujours dans l'exemple d'un patient dans le coma. Les médecins s'interrogent sur des décisions irréversibles, dans le cas où ce terme de « coma » impliquerait que le patient n'ait plus aucun espoir de réveil. Dès lors, il vaut mieux que la méthode oublie des patients réellement dans le coma, plutôt que de prédire qu'un patient conscient est en réalité inconscient. Nous avons représenté à la FIGURE 4.10 les deux possibilités que nous pouvons avoir, pour une meilleure compréhension.

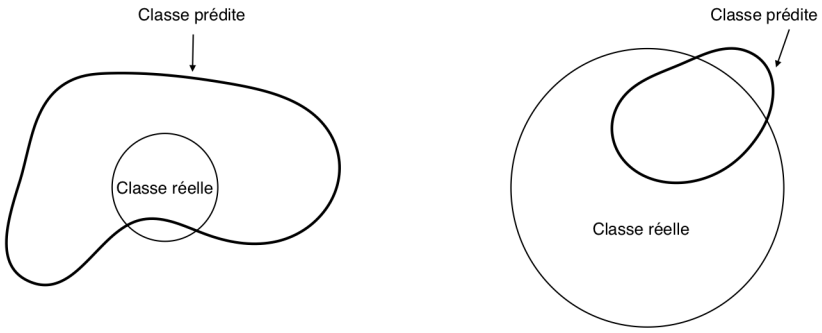


FIGURE 4.10 – L'image de gauche représente le cas d'une méthode dite de « détection ». En effet, elle a tendance à inclure dans une classe plus d'observations afin de maximiser le fait que tous les patients appartenant effectivement à cette classe seront prédits comme tels. A droite, nous avons une représentation schématique de ce qu'est une méthode dite « prudente ». En effet, cette méthode, lorsqu'elle prédit que des patients sont dans une classe, essaie de se tromper le moins possible. Dès lors, elle se contentera de sélectionner une petite population afin de minimiser le taux d'erreur de prédiction.

Nous allons discuter les deux types de méthodes, selon la classe considérée, afin d'attribuer la meilleure de nos méthodes de classification à chaque classe. En effet, nous serons certainement amenés à sélectionner des méthodes différentes selon les cas considérés.

Discussion de la classe « Conscient »

Dans un premier temps, regardons quelle classification serait la plus adaptée pour avoir une méthode de détection. Dans un deuxième temps, nous regarderons la plus adaptée à une méthode prudente. La méthode de détection a tendance à englober des patients qui ne sont pas conscients dans la classe « Conscient » en plus de ceux qui sont bel et bien conscients. Dès lors, cette méthode possède idéalement de nombreux vrais positifs, et un nombre de faux positifs qui peut être important, mais dont on ne se soucie pas réellement.

Par conséquent, une méthode de détection est une méthode qui peut ne pas être très précise. En termes de spécificité et sensibilité, une méthode de détection est une méthode pouvant être peu spécifique. Dans l'idéal cependant, une méthode de détection, même si le nombre de faux positifs importe peu, détectera tout de même un maximum de vrais positifs.

Parmi les classifications que nous avons réalisées, nous regardons celle qui possède la plus grande sensibilité. La méthode de détection de la classe « Conscient » est la quatrième classification. En effet, avec $61,1\%$ de sensibilité, c'est cette méthode qui permet de détecter au mieux les patients conscients.

Concernant la méthode prudente, il s'agit d'une méthode qui peut prédire peu de vrais positifs, pour autant qu'elle prédise un minimum de faux positifs. Dès lors, au plus la précision de la classe est élevée, au plus la méthode sera prudente. A nouveau, c'est la quatrième classification qui tire son épingle du jeu, avec $61,1\%$ de précision concernant la classe « Conscient ». Pour cette première classe, c'est donc la quatrième classification qui donne les meilleurs résultats pour les deux types de méthodes.

Nous avons donc sélectionné des modèles en fonction du type de méthode que nous voulons avoir. De manière intuitive, c'est la méthode de détection qui sera la plus utile pour la classe « Conscient ». Nous voulons en effet détecter à tout prix qu'un patient est conscient, quitte à faire des erreurs, plutôt que le détecter inconscient et prendre des décisions irréversibles.

Discussion de la classe « Moyennement endormi »

Nous allons procéder de la même manière que précédemment pour sélectionner les méthodes dites de « détection » et « prudente » pour la classe des patients « Moyennement endormis ». Ainsi, pour cette première, nous regardons à nouveau le pourcentage de la sensibilité donné par la matrice de confusion. Dans ce cas-ci, c'est la cinquième classification qui possède le pourcentage le plus élevé, avec $55,6\%$.

En ce qui concerne le choix de la méthode prudente, nous devons à nouveau regarder la précision de la classe « Moyennement endormi » lors de chaque classification. Il s'agit dans ce cas-ci de la cinquième classification. En effet, nous avons obtenu $62,5\%$ de précision pour cette deuxième classe lors de la classification.

Discussion de la classe « Profondément endormi »

Pour déterminer la méthode dite de « détection » pour les patients de cette classe « Profondément endormi », nous nous référons au pourcentage de la troisième colonne de la matrice de confusion associée à chaque classification. Les trois dernières classifications possèdent un pourcentage de détection de $72,2\%$. Dès lors, s'il faut en sélectionner une des trois, nous choisirions la cinquième classification, puisqu'elle possède une *accuracy* plus élevée que les autres.

Regardons à présent quelle classification peut être associée à la méthode prudente de cette troisième classe. Pour ce faire, regardons à nouveau les pourcentages de précision de la classe dans chacun des tests et sélectionnons celui qui possède la précision la plus élevée. Il s'agit de la troisième classification, avec un pourcentage de précision de $81,2\%$.

Discussion de la classe « Phase de réveil »

Les classifications qui possèdent la plus grande sensibilité pour la dernière classe sont, avec $66,7\%$, la première et la cinquième. Dès lors, ce sont ces deux méthodes qui sont les plus aptes à détecter qu'un patient est en phase de réveil. Comme pour la classe précédente, nous choisissons cependant la cinquième classification puisque celle-ci possède, à même sensibilité, une *accuracy* plus élevée.

C'est également cette dernière classification qui possède la plus grande précision pour la classe « Phase de réveil », avec $70,6\%$. Dès lors, c'est cette méthode qui sera dite prudente.

Récapitulatif des choix

Comme nous venons de le voir, associer la méthode de classification la plus appropriée pour la prédiction d'une classe est à nouveau un choix délicat. En effet, il n'existe pas de règle spécifique qui impose la sélection d'une méthode plutôt qu'une autre pour prédire au mieux chaque classe. De plus, comme nous l'avons déjà rappelé, la classification basée sur les *Extra-Trees* possède une part d'aléatoire qui se reflète par une légère variance des résultats. Dans la Table 4.6, nous reprenons les différents choix qui ont été réalisés pour la prédiction des différentes classes.

	Méthode de détection	Méthode prudente
« Conscient »	Classification 4	Classification 4
« Moyennement endormi »	Classification 5	Classification 5
« Profondément endormi »	Classification 5	Classification 3
« Phase de réveil »	Classification 5	Classification 5

Table 4.6 – Table récapitulative des choix de classification.

Nous remarquons que la cinquième classification a été sélectionnée le plus de fois, ce qui reflète ses bons résultats énoncés précédemment.

4.4 Agrégation des classes

Comme illustré précédemment, il n'est pas évident de prédire simultanément les quatre classes considérées. En effet, les classes des états de conscience intermédiaires sont, dans la majorité des classifications retenues, les deux cas les plus difficiles à prédire. Dès lors, nous avons décidé de les agréger dans le but d'avoir de meilleurs résultats. La deuxième motivation de ce choix est que ces états sont assez proches de par la nature des données. Pour rappel, les patients ont été soumis à un produit anesthésiant afin de simuler le coma. Dès lors, comme nous l'avons souligné au début de ce chapitre, les mesures des états intermédiaires sont moins précises que les états extrêmes, puisque le sédatif agit rapidement et donc il est difficile d'obtenir les mesures dans un même état pour tous les patients. De plus, dans ces deux états, les patients sont éveillés mais pas totalement conscients.

Nous avons décidé de réaliser cette agrégation de classes sur la quatrième classification, c'est-à-dire que nous relançons la classification en considérant les mêmes variables, mais en agrégeant les deux classes dont il est question. En effet, elle n'a pas été retenue dans l'analyse précédente pour les deux classes en question. De plus, ses résultats concernant les états conscients et d'endormissement profond sont plus satisfaisants que les deux premières méthodes.

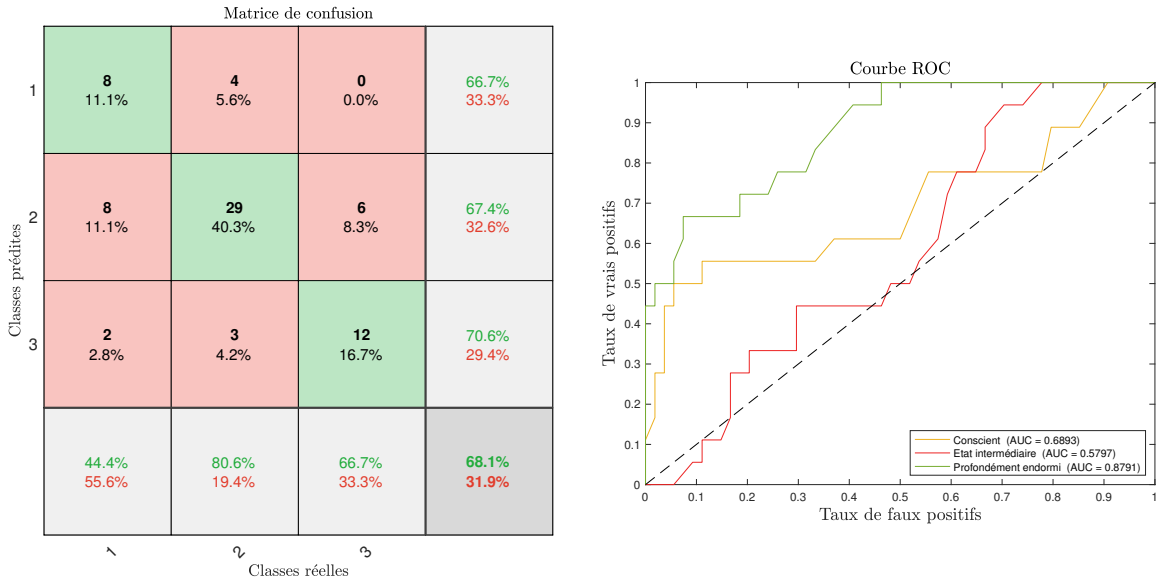


FIGURE 4.11 – Matrice de confusion et courbe ROC pour la classification réalisée sur les mêmes variables que la quatrième méthode. Les classes intermédiaires ont été agrégées en une classe « Etat intermédiaire », et se trouvant en deuxième position sur la matrice de confusion.

Nous pouvons observer les différents résultats de cette classification sur la FIGURE 4.11. Remarquons tout d'abord que la méthode a encore moins tendance qu'avant à confondre des patients conscients à des patients profondément endormis et vice-versa. Par rapport à la classification de la FIGURE 4.7, la sensibilité de la classe « Conscient » a diminué, bien que la sensibilité

globale ait augmenté. En ce qui concerne la précision des classes similaires aux deux classifications, celle-ci a été légèrement améliorée, passant de $61,1\%$ à $66,7\%$ pour les patients conscients et de 65% à $70,6\%$ pour les patients profondément endormis. En ce qui concerne cette nouvelle classe comprenant les états intermédiaires, elle possède de très bons résultats. La sensibilité de cette classe atteint les $80,6\%$ et sa précision les $67,4\%$.

Comme pour les analyses précédentes, nous pouvons tracer le tableau de confusion et calculer les spécificités des classes. Celui-ci se trouve à la TABLE 4.7.

	Conscient	¬Conscient	Interméd.	¬Interméd.	Profond	¬Profond
Test positif	8	4	29	14	12	5
Test négatif	10	50	7	22	6	49

TABLE 4.7 – Tables de confusion pour les trois classes de la classification effectuée sur les ensembles agrégés.

Dès lors, nous avons les spécificités

$$Sp_C = \frac{50}{54} = 92,59\%, \quad Sp_M = \frac{22}{36} = 90,74\% \quad \text{et} \quad Sp_R = \frac{49}{54} = 77,36\%,$$

qui sont meilleures que précédemment, pour les deux classes en commun. Concernant la courbe ROC, nous aurions pu nous attendre à un meilleur résultat pour la classe agrégée. En effet, les deux classes étant regroupées, la méthode a moins de possibilités pour disperser ses prédictions. Nous pensions donc que la méthode aurait été meilleure, mais ce n'est pas le cas ; nous avons obtenu une AUC de $0,5797$. La courbe ROC de la classe des patients profondément endormis est cependant meilleure que lors de la quatrième classification.

L'agrégation des deux classes d'états intermédiaires permet donc bien d'améliorer la prédiction de ces deux classes. Nous ne savons cependant pas les distinguer sans effectuer de tests complémentaires. Le but de cette section étant d'observer les changements que causent une agrégation de classes sur les résultats de la classification, nous n'irons pas plus loin dans le domaine, même si nous pouvons réaliser le même processus de réflexion quant aux sélections de variables.

4.5 Comparaison des résultats avec d'autres méthodes de classification

Jusqu'ici, nous n'avons réalisé les classifications qu'à l'aide de la méthode *Extra-Trees*. Pour rappel, celle-ci introduit de l'aléatoire lors de la sélection des variables qui servent à la subdivision des feuilles de chaque arbre de la forêt. Malgré le fait que cette composante aléatoire est censée réduire la variance des résultats, nous observons des changements d'une exécution à l'autre. Nous avons donc décidé d'effectuer deux classifications supplémentaires. Le but premier

est d'utiliser une méthode où les résultats seraient identiques d'une exécution à l'autre, mais également de voir que cet aléatoire améliore bien les résultats de la classification.

Pour ce faire, nous utilisons une méthode d'arbre classique, et la régression logistique, toutes deux présentées au chapitre 2.2. Le but étant une comparaison des résultats, nous réalisons deux classifications naïves, c'est-à-dire que nous utilisons les mêmes variables que précédemment ; nous ne recommençons pas un processus de sélection. Notons tout de même que ces variables ont été ajustées à la méthode des *Extra-Trees*, il sera donc normal que les résultats soient moins bons dans ces comparaisons.

Nous utilisons les variables de la cinquième classification, car cette dernière est la plus performante, comme illustré dans la TABLE 4.6. Notons que nous réalisons les classifications sur les quatre classes initiales et non pas sur les classes agrégées.

4.5.1 Arbre de décision

L'arbre de décision que nous utilisons ici est celui fourni par le logiciel Matlab, à l'aide de la fonction `fitctree`. Nous effectuerons à nouveau une méthode de *Cross-Validation*, afin d'entraîner au mieux notre modèle.

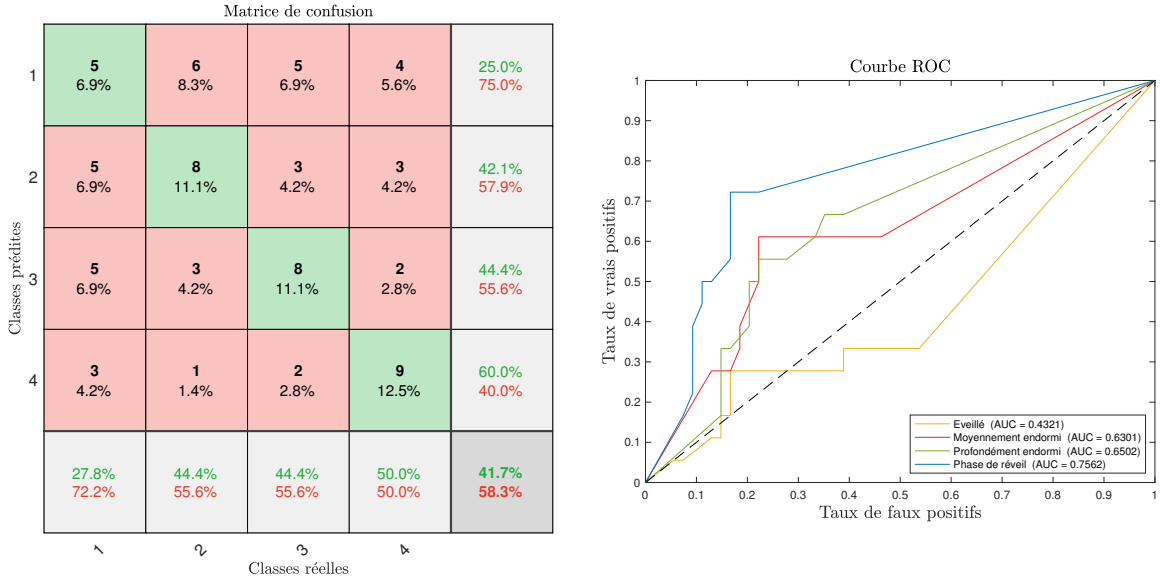


FIGURE 4.12 – Matrice de confusion et courbes ROC de la classification effectuée par la méthode des arbres de décision.

Lorsque nous analysons la matrice de confusion de la FIGURE 4.12 et que nous la comparons à la cinquième classification réalisée à l'aide de la méthode des *Extra-Trees*, nous pouvons conclure que les résultats sont moins bons. En effet, l'*accuracy* de la méthode n'est que de $41,7\%$

contre $59,7\%$ pour l'équivalent en *Extra-Trees*. De plus, aucune des sensibilités de la méthode ne parvient à faire mieux que pour les *Extra-Trees*. La constatation est la même concernant les précisions.

Comme nous l'avons vu lors de sa présentation, une courbe ROC sert essentiellement à comparer des méthodes entre-elles. La courbe de l'arbre associée à la classe des patients conscients est relativement médiocre, puisqu'elle se situe sous la diagonale et possède une AUC de $0,4321$ qui est inférieur aux $0,5$ d'une méthode purement aléatoire. Concernant les autres courbes, elles sont satisfaisantes de par leur allure et leur AUC, mais elles sont tout de même moins bonnes que celles de la cinquième classification réalisée à l'aide des *Extra-Trees*. Nous pouvons donc conclure, au vu de la matrice de confusion et de la courbe ROC, que cette méthode d'arbre est moins performante que la méthode des *Extra-Trees*. Nous pourrions tenter d'expliquer cette diminution de performance par le fait que nous ne considérons qu'un seul arbre de décision et non pas une forêt, ce qui pourrait potentiellement améliorer les résultats.

4.5.2 Régression logistique

Afin de comparer les résultats de la classification issue des *Extra-Trees* à une régression logistique, nous réalisons à nouveau une *cross-validation* avec la fonction de régression logistique fournie par Matlab, `mnrfit`.

Les différents résultats de cette classification se trouvent à la FIGURE 4.13. L'*accuracy* de cette méthode est moins bonne que la méthode utilisant les *Extra-Trees*. Lorsque nous regardons les éléments de la matrice de confusion, nous pouvons remarquer que la sensibilité de la quatrième classe, celle des patients en phase de réveil, est identique à celle de la cinquième classification *Extra-Trees*. La précision de cette classe est même meilleure avec la régression logistique, puisqu'elle est de $70,6\%$ contre $63,2\%$ pour la classification *Extra-Trees*. Les autres sensibilités et précisions sont par contre meilleures pour la classification utilisant les *Extra-Trees*.

Dans le but de calculer et comparer les valeurs de spécificité de chacune des classes, nous reprenons à la TABLE 4.8 les différentes valeurs de la table de confusion.

La précision de la classe des patients profondément endormis est plus élevée par rapport à la cinquième classification *Extra-Trees*, avec $66,7\%$ contre $61,9\%$. La spécificité de la régression logistique est également meilleure pour cette classe. En effet, en se basant sur les résultats des tables de confusion, nous obtenons les spécificités,

$$Sp_C = \frac{44}{54} = 81,48\%, \quad Sp_M = \frac{41}{54} = 75,93\%, \quad Sp_P = \frac{48}{54} = 88,89\% \quad \text{et} \quad Sp_R = \frac{47}{54} = 87,03\%.$$

Concernant les courbes ROC, elles sont meilleures pour la classe de patients conscients et profondément endormis. Cependant, au vu des résultats de la matrice de confusion, nous pouvons dire que la méthode des *Extra-Trees* est globalement meilleure pour le jeu de variables considéré.

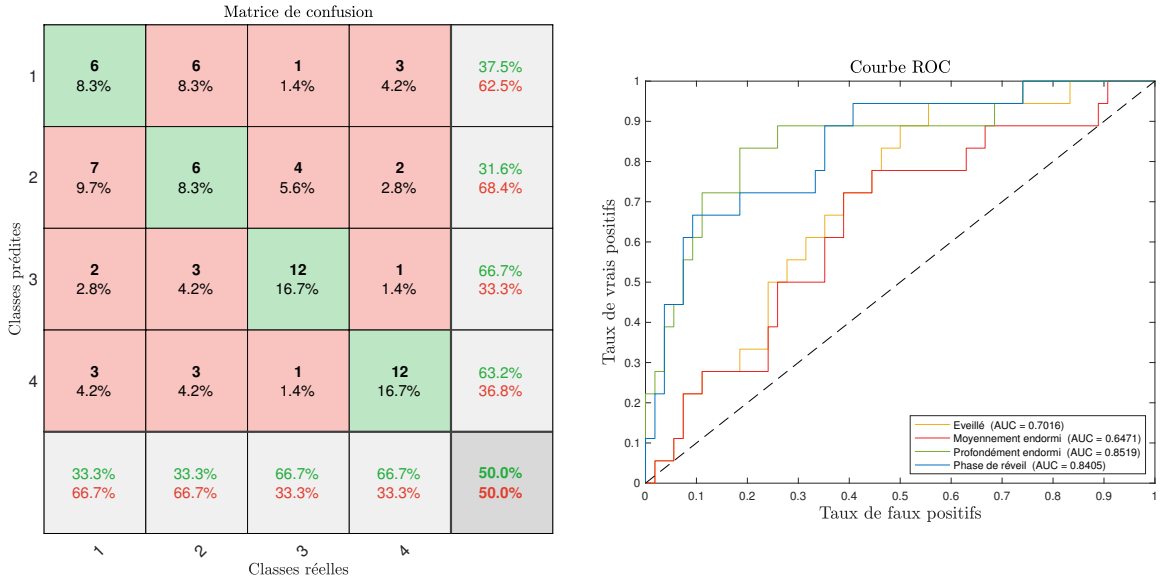


FIGURE 4.13 – Matrice de confusion et courbes ROC de la classification effectuée en utilisant la régression logistique.

	Conscient	¬Conscient	Moyen	¬Moyen
Test positif	6	10	6	13
Test négatif	12	44	12	41

	Profond	¬Profond	Réveil	¬Réveil
Test positif	12	6	12	7
Test négatif	6	48	6	47

TABLE 4.8 – Tables de confusion associées à chacune des classes considérées, pour la classification à l'aide de la régression logistique.

4.6 Conclusion

Dans ce chapitre, nous avons réalisé différentes classifications, afin de prédire au mieux l'état de conscience d'un patient. Nous avons tout d'abord utilisé les variables sélectionnées dans le chapitre précédent, afin de choisir un des trois jeux de données, avant de réaliser d'autres classifications sur les données *reduced GS*. Les différentes classifications successives ont amélioré les différents modèles. Pour obtenir ces améliorations, nous avons dû modifier les ensembles de variables utilisées pour les classifications. Cette modification peut aller de l'ajout de nouvelles variables, au retrait d'autres, en passant par l'ajout de variables déjà retirées. Nous n'avons ce-

pendant obtenu aucune classification permettant de prédire presque parfaitement chaque classe, c'est pourquoi nous avons sélectionné les meilleures classifications selon le type de test que nous voudrions obtenir. Cette sélection est rappelée à la TABLE 4.6.

Les classifications que nous avons réalisées ont confirmé notre intuition : les classes des états intermédiaires sont généralement les plus difficiles à prédire. Dès lors, nous avons réalisé une nouvelle classification, sur base des variables de la quatrième méthode, en agrégeant ces deux états intermédiaires en une seule classe. Les résultats sont, malgré une diminution de la qualité de prédiction de la première classe, bien meilleurs que lorsque nous avions deux classes séparées. Nous avons en effet réussi à atteindre $80,6\%$ de détection et $67,4\%$ de précision pour cette classe.

Dans le but d'avoir une comparaison possible avec une méthode non-aléatoire, mais également dans le but de vérifier que l'aléatoire des *Extra-Trees* améliore bien les résultats, nous avons réalisé deux autres classifications à l'aide de deux autres méthodes : l'arbre de décision et la régression logistique. Les résultats de ces deux classifications sont moins bons que ceux de l'*Extra-Trees*, à l'exception de la classe des patients en phase de réveil prédite par la régression logistique.

Conclusion et perspectives

L'objectif de ce mémoire était l'introduction d'une nouvelle technique de détermination de l'état de conscience d'un patient. Nous avons donc développé une méthode de classification spectrale, qui exploite les données IRMf du patient, afin de prédire au mieux son état de conscience.

Le premier chapitre s'intéressait à la récolte des données d'imagerie médicale. L'imagerie par résonance magnétique fonctionnelle est une technique très répandue qui permet d'obtenir des informations sur l'activité cérébrale. Celle-ci mesure les variations hémodynamiques du cerveau. L'IRMf fournit les mesures sous forme de signal BOLD, *Blood Oxygen Level Dependant*. Ce signal peut être simulé grâce au modèle *Dynamic Causal Modelling*, qui modélise les interactions entre les différents neurones du cerveau. Il se base sur différentes équations d'état et permet de simuler les influences causées par différents stimuli sur le réseau considéré.

Le point crucial de ce mémoire était le lien qui existe entre le spectre du réseau considéré, le cerveau, et le spectre de l'opérateur de Koopman. La théorie de l'opérateur de Koopman permet l'introduction des modes de Koopman, qui peuvent être calculés grâce à l'algorithme de décomposition en modes dynamiques, sur base de la dynamique du cerveau. Le second chapitre se consacrait également aux méthodes de classification telles que la méthode des *Extra-Trees*, ainsi qu'aux mesures de la qualité de ces classifications. La méthode de classification *Extra-Trees* consiste à créer une forêt d'arbres de décision, dans lesquels les critères de coupure des feuilles sont sélectionnés parmi un sous-ensemble de variables sélectionnées aléatoirement.

Le troisième chapitre consistait en un entrainement de notre méthode. En effet, grâce au modèle DCM, nous avons simulé des données IRMf pour une dizaine de patients. A ceux-ci nous avons fait correspondre deux états de conscience différents. Nous avons ensuite appliqué l'algorithme de décomposition en modes dynamiques afin d'obtenir le spectre de l'opérateur de Koopman. Nous avons sélectionné des variables dans ce spectre, sur lesquelles nous avons effectué une classification *Extra-Trees*. Les premiers résultats n'étaient pas très concluants, c'est pourquoi nous avons amélioré la classification en modifiant les variables sélectionnées, mais également en sélectionnant d'autres variables spectrales. Notre modèle final, qui avait une *accuracy* de 90%, nous a donné des résultats satisfaisants pour les simulations.

Pour terminer, le quatrième et dernier chapitre s'intéressait aux données réelles de l'IRMf. Nous y avons présenté les trois jeux de données, chacun d'eux ayant subi une phase de prétraitement différente. Nous avons commencé par réaliser une classification sur base des variables sélectionnées dans le chapitre précédent, afin de choisir le jeu de données qui donne de meilleurs résultats. Il s'agit des données sur lesquelles le signal global a été retiré, c'est-à-dire les données *reduced GS*. Nous avons réalisé au total cinq classifications qui ont donné de bons résultats,

bien qu'aucune ne nous fournisse des résultats satisfaisants pour tous les états de conscience. C'est pourquoi nous avons associé à chaque classe la méthode la plus adéquate, selon que nous voulions une méthode de précision ou une méthode de détection. Au cours de nos analyses, nous avons remarqué que les classes les plus difficiles à prédire sont celles des états intermédiaires, c'est-à-dire l'état « Moyennement endormi » et l'état « Phase de réveil ». Étant des états de transition entre la conscience et le sommeil profond, les mesures ne sont en effet pas aussi catégoriques que pour les deux autres classes. Dès lors, nous avons décidé d'agréger ces deux classes en une afin d'améliorer cette prédiction. Grâce à cette nouvelle classe, nous avons obtenu d'excellents résultats, à savoir $80,6\%$ de sensibilité et $67,4\%$ de précision. Dans le but de réaliser une brève comparaison entre la méthode *Extra-Trees* et des méthodes qui ne font pas intervenir d'aléatoire, nous avons terminé ce chapitre par une classification à l'aide d'un arbre de décision et la régression logistique. Les résultats sont inférieurs à ceux obtenus via les *Extra-Trees*.

Au terme de ce mémoire, nous avons proposé une méthode de détection de l'état de conscience d'un patient, alternative et compétitive par rapport à celles existantes. Cependant, la difficulté de cette méthode, et une des contributions de ce travail, réside dans le choix des variables nécessaires à la classification. En effet, aucune méthode ne nous permet de connaître à l'avance quelles sont les variables pertinentes.

Bien que nous soyons arrivés à des résultats concluants, nous avons mis en évidence, au cours de notre recherche, quelques pistes d'améliorations. Nous avons sélectionné la méthode des *Extra-Trees*, indépendamment de nos jeux de données. Dès lors, il serait intéressant d'étudier la possibilité d'utiliser une méthode plus adaptée à ceux-ci. L'article [14] montre en effet certains jeux de données dont l'erreur varie selon le choix des paramètres. Une perspective qui pourrait s'avérer également intéressante serait de comparer les résultats de notre méthode avec ceux d'une méthode basée sur les corrélations (voir par exemple [16]).

Bibliographie

[1] Andreelli F., Mosbah H., *IRM fonctionnelle cérébrale : les principes*, Médecine des Maladies Métaboliques, 8(1), 13-19, 2014.

[2] Arbabi H., *Introduction to Koopman operator theory of dynamical systems*, https://engineering.ucsb.edu/~harbabi/research/KoopmanIntro.pdf, 07 mai 2018.

[3] Arlot S., *Classification supervisée : des algorithmes et leur calibration automatique*, https://www.math.u-psud.fr/~arlot/enseign/2009Centrale/cours-classif.pdf, 01 mai 2018.

[4] Barthelemy J., *Cours : Analyse multivariée et introduction aux logiciels statistiques*, SMATM102, Université de Namur, 2017.

[5] Benzaki Y., *Logistic Regression pour Machine Learning - Une introduction simple*, https://mrmint.fr/logistic-regression-machine-learning-introduction-simple, 19 mai 2018.

[6] Bernard S., *Forêts Aléatoires : De l'Analyse des Mécanismes de Fonctionnement à la Construction Dynamique*, Interface homme-machine, Université de Rouen, 2009.

[7] Bruno M.A. et al., *Quelle vie après le ocked-in syndrome ?*, Revue médicale de Liège, 63(5-6), 445-451, 2008.

[8] Chavent M., *Apprentissage supervisé : Introduction*, http://www.math.u-bordeaux.fr/~mchave100p/wordpress/wp-content/uploads/2013/10/App_C1.pdf, 12 mai 2018.

[9] Crossley N.A. et al., *Cognitive relevance of the community structure of the human brain functional coactivation network*, NeuroScience, 110(28), 11583–11588, 2013.

[10] De Schryver R., *Dynamic Mode Decomposition n Hydrodynamics : Influence of Noise& Missing Data*, Master of Science in Civil Engineering, Université de Gand, Gand, 2016.

[11] Depiereux E. et al., *Pratique des biostatistiques*, http://webapps.fundp.ac.be/biostats/biostat/modules/module35/page3.html, 11 mai 2018.

[12] Frénay B., *Cours : Introduction to Data Science : Part1. Basic Concepts and Decision Trees*, SMATM203, Uiversité de Namur, 2018.

[13] Friston K.J., Harrisson L., Penny W., *Dynamic causal modelling*, NeuroImage, 19(4), 1273-1302, 2003.

[14] Geurts P., Ernst D., Wehenkel L., *Extremely randomized trees*, Machine Learning, 63(1), 3-42, 2006.

[15] Glaser D.E.., et. al., *Haemodynamic modelling*, http://www.fil.ion.ucl.ac.uk/spm/doc/books/hbf2/pdfs/Ch11.pdf, 4 avril 2018.

[16] Heine L. et al., *Resting state networks and consciousness : Alterations of multiple resting stat network connectivity in physiological, pharmacological, and pathological consciousness states*, Frontiers in Psychology, 3(295), 1-12, 2012.

[17] Inconnu, *L'imagerie par résonance magnétique fonctionnelle (IRMf)*, http://www.reflexions.uliege.be/cms/c_7613/l-imagerie-par-resonance-magnetique-fonctionnelle-irmf, 6 avril 2018.

[18] Laureys S., Pellas F., Van Eeckout P., *Le Locked-In Syndrome*, https://www.alis-asso.fr/wp-content/uploads/2014/05/Publication_fcfa1.pdf, 27 mai 2018.

[19] Marreiros A. et al., *Dynamic causal modeling*, http://www.scholarpedia.org/article/Dynamic_causal_modeling, 01 avril 2017.

[20] Mathworks, *Aide Matlab : plotconfusion*, https://nl.mathworks.com/help/nnet/ref/plotconfusion.html, 11 mai 2018.

[21] Mathworks, *Aide Matlab : perfcurve*, https://nl.mathworks.com/help/stats/perfcurve.html, 12 mai 2018.

[22] Mauroy A., Hendrickx J., *Spectral Identification of Networks Using Sparse Measurements*, SIAM Journal on Applied Dynamical Systems, 16(1), 479-513, 2017.

[23] Mauroy A., *Cours : Méthodes avancées pour les systèmes non-linéaires*, SMATM227, Université de Namur, 2017.

[24] Mezić I., *Analysis of Fluid Flows via Spectral Properties of the Koopman Operator*, Annual Review of Fluid Mechanics, 45, 357-378, 2013.

[25] Nendas M., Perrier A., *Sensibilité, spécificité, valeur prédictive positive et valeur prédictive négative d'un test diagnostique*, http://www.em-consulte.com/showarticlefile/144227/index.pdf, 11 mai 2018.

[26] Oxford, *Introduction to FMRI*, https://www.ndcn.ox.ac.uk/divisions/fmrib/what-is-fmri/introduction-to-fmri, 20 mars 2017.

[27] Stephan K. et al., *Comparing hemodynamic models with DCM*, NeuroImage, 38, 387-401, 2007.

[28] Tu J. et al., *On Dynamic Mode Decomposition : Theory and Applications*, Journal of Computational Dynamics, 1(2), 391-421, 2014.

[29] Walber, *Illustration : Precision - Recall*, https://en.wikipedia.org/wiki/Precision_and_recall, 15 mai 2018.

Annexe A

Simulations

Dans cette annexe se retrouvent les différents résultats mentionnés dans le chapitre 3. Il s'agit essentiellement des valeurs propres de la matrice d'adjacence des deux réseaux simulés, ainsi que les différents résultats énoncés concernant les simulations.

A.1 Définition des réseaux simulés

Nous reprenons dans cette section les valeurs propres de la matrice d'adjacence des deux réseaux, ainsi que les matrices B et C, modélisant respectivement les changements dans les couplages dûs à l'entrée u et les influences directes de cette entrée sur le réseau.

A.1.1 Valeurs propres

Les valeurs propres de la matrice d'adjacence des deux réseaux simulés sont

$$eig(A_1) = \begin{pmatrix} -0,5000 \\ -1,4269 \\ -1,3166 + 0,0524i \\ -1,3166 - 0,0524i \\ -1,1872 + 0,1458i \\ -1,1872 - 0,1458i \\ -1,1414 + 0,1096i \\ -1,1414 - 0,1096i \\ -1,0727 \\ -1,1594 \end{pmatrix} \quad \text{et} \quad eig(A_2) = \begin{pmatrix} -1,0013 + 0,2583i \\ -1,0013 - 0,2583i \\ -0,3062 \\ -0,4403 \\ -0,5545 + 0,1269i \\ -0,5545 - 0,1269i \\ -0,8326 \\ -0,7631 \\ -0,6466 \\ -0,7623 \end{pmatrix}.$$

Nous pouvons observer que, quel que soit le réseau considéré, le partie réelle de chaque valeur propre est négative, ce qui garantit la stabilité des deux réseaux.

A.1.2 Influences de l'entrée u

Nous reprenons les deux matrices des équations d'état neuronal qui définissent les changements dûs à l'entrée u. Il s'agit de la matrice B, qui introduit les changements de couplages, et la matrice C qui introduit l'influence directe de l'entrée. Ces deux matrices, pour le premier réseau, sont

$$B = \begin{bmatrix} 0 & 0,1427 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0,1315 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ et } C = 0,01 \times \begin{bmatrix} 0,8199 \\ 0,9545 \\ 0,4784 \\ 0,7478 \\ 0,7004 \\ 0,3317 \\ 0,1237 \\ 0,9127 \\ 0,8420 \\ 0,9525 \end{bmatrix},$$

et pour le second,

$$B = \begin{bmatrix} 0 & 0,1427 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0,1315 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,4709 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ et } C = 0,01 \times \begin{bmatrix} 0,9055 \\ 1,0194 \\ 0,5029 \\ 0,8020 \\ 0,7894 \\ 0,4198 \\ 0,1924 \\ 0,9633 \\ 0,9043 \\ 0,9676 \end{bmatrix}.$$

A.2 Résultats des simulations

Les résultats des classifications sur les simulations énoncés dans le chapitre 3 sont repris ci-dessous. Dans la FIGURE A.1, nous avons les différents résultats de la deuxième classification réalisée.

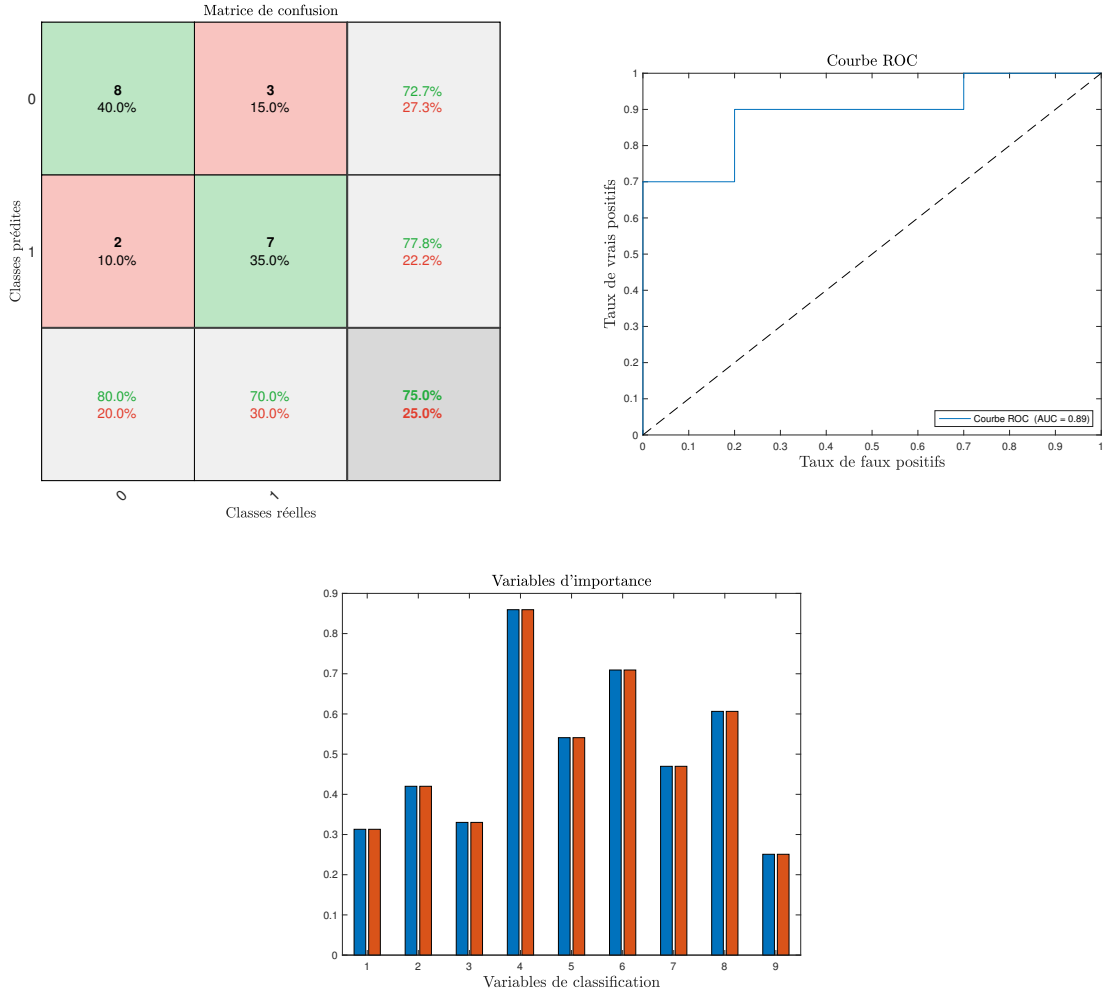


FIGURE A.1 – Matrice de confusion (à gauche) et ROC Curve (à droite) et variables d'importance (en bas) relatives à la deuxième classification.

Nous avons placé les résultats de la troisième classification des simulations dans la FIGURE A.2.

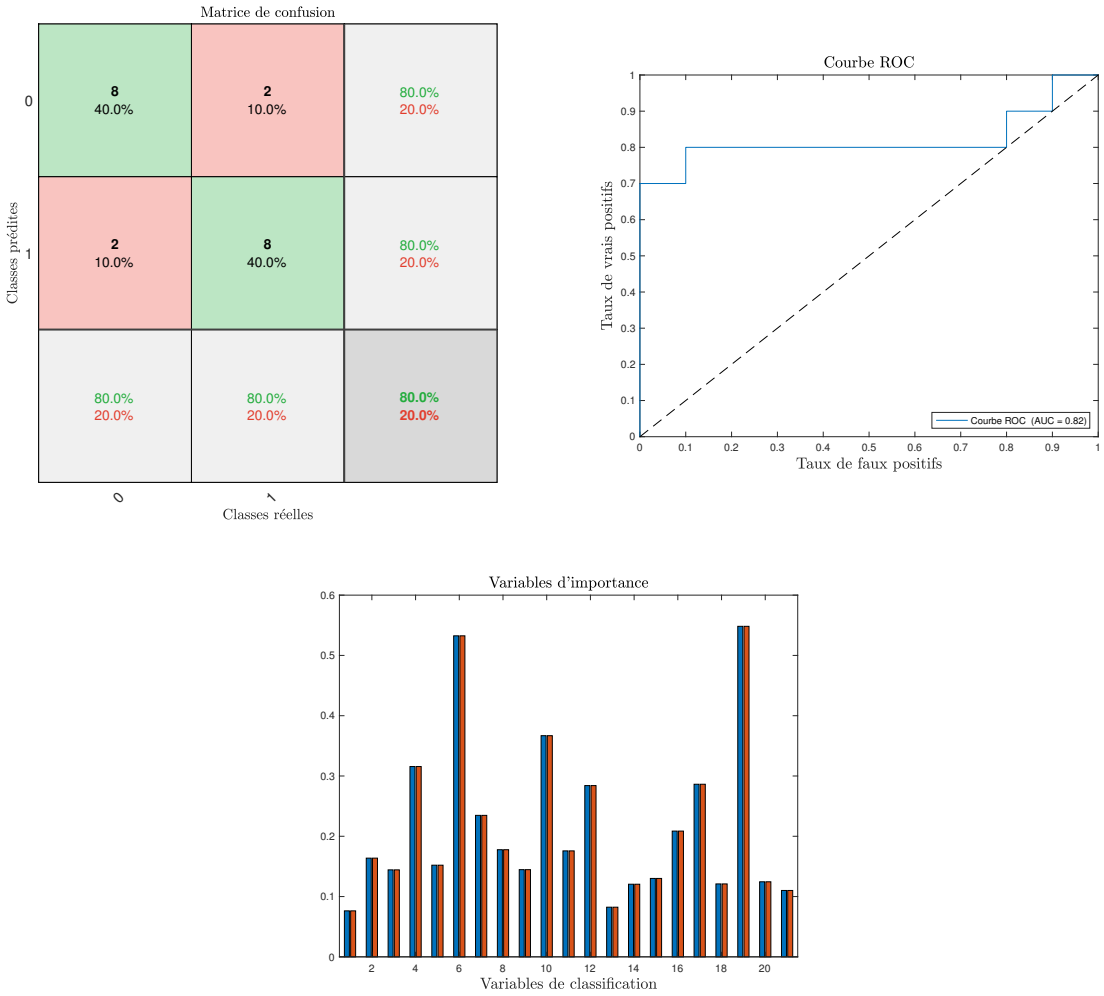


FIGURE A.2 – Matrice de confusion (à gauche) et ROC Curve (à droite) et variables d'importance (en bas) relatives à la troisième classification.

A.3 Résultats des données

Nous reprenons dans la FIGURE A.3 les variables d'importance concernant la cinquième classification sur les données IRMf.

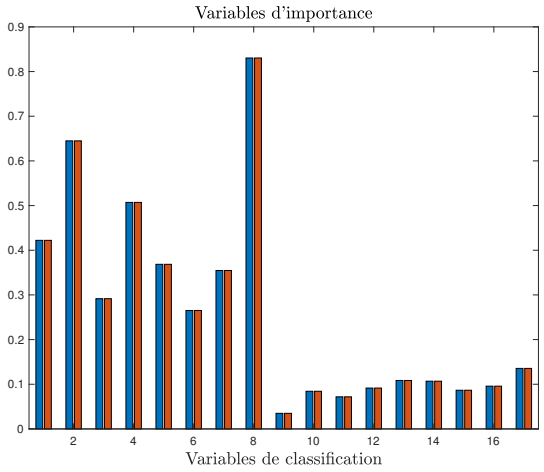


FIGURE A.3 – Matrice de confusion (à gauche) et ROC Curve (à droite) et variables d'importance (en bas)relatives à la troisième classification.

Annexe B

Codes

Dans cette annexe, nous reprenons les différents codes que nous avons utilisés afin d'obtenir les résultats présentés dans ce travail. Nous commençons par reprendre le code de l'algorithme DMD, puis ceux qui nous ont servis pour les simulations, avant de mettre ceux relatifs aux données réelles. Notons tout de même que la méthode des *Extra-Trees* a été codée par Pierre Geurts, de l'Université de Liège, et les scripts sont disponibles sur le site internet, `http://www.montefiore.ulg.ac.be/~geurts/Software.html`. Notons que l'écriture de ces codes n'a pas été optimisée comme elle aurait pu l'être.

B.1 Décomposition en modes dynamiques

```
function [modes,lambda,rescaling] = DMD(Z,deltat,tri)
% INPUTS:
% Z = matrice dans laquelle les vecteurs x_i sont dans la 1e colonne
%           et les vecteurs y_i dans la 2e colonne
% deltat = période d'échantillonage des données
% tri = choix de tri:
%           tri = 1: outputs triés selon la domiance des valeurs propres
%                   (la plus dominante à doite)
%           tri = 2: Outputs triés selon la grandeur du rescaling
%                   (le plus grand à droite)
% OUTPUTS:
% lambda = vecteur des valeurs propres continues DMD
%           triées selon "tri"
% modes = matrice dont les colonnes correspondent aux modes de DMD associés
%           aux valeurs propres lambda, triés selon "tri"
% rescaling = paramètres de scaling, triés selon "tri"

y_0 = Z(:,2);
% ___ Etape 1: ___ %
X = Z(:,1:size(Z,2)-1);
Y = Z(:,2:size(Z,2));

% ___ Etape 2: ___ %
[U,S,V] = svd(X,'econ');

```

```
% ___ Etape 3: ___%
A = U'*Y*V*inv(S);

% ___ Etape 4: ___ %
[VeP,ValP,W] = eig(A);

% ___ Etape 5: ___ %
for j=1:length(ValP)
   lambda(j) = log(ValP(j,j))/(deltat);
   for l=1:size(VeP,1)
   modes(l,j) = U(l,:)*VeP(:,j);
   end
end

% ___ Etape 6: Rescaling ___ %
rescaling = inv(ValP)*inv(modes)*y_0;

switch tri
    case 1 % Par dominance des valeurs propres
        [lambda,modes,rescaling] = tri_insert(lambda,modes,rescaling',size(lambda,2));
    case 2 % Par importance du rescaling
        [rescaling,modes,lambda] = ...
            tri_insert(rescaling',modes,lambda,size(rescaling',2));
end
end
```

```
function [tableau,tab1,tab2] = tri_insert(tableau,tab1,tab2,n)
% tableau = tableau à trier (à 1 dimension)
% tab1 = tableau en 2D qui suit le tri de tableau par colonne
% tab2 = tableau en 2D qui suit le tri de tableau par colonne
% n = nombre d'éléments du tableau

j = 2;
while j<=n
    i = 1;
    tmp = tableau(j);
    tmp_tab1 = tab1(:,j);
    tmp_tab2 = tab2(:,j);
    while i <= j-1 && real(tmp) >= real(tableau(i))
        i = i+1;
    end
    for k = j:-1:i+1
        tableau(k) = tableau(k-1);
        tab1(:,k) = tab1(:,k-1);
        tab2(:,k) = tab2(:,k-1);
    end
    tableau(i) = tmp;
    tab1(:,i) = tmp_tab1;
    tab2(:,i) = tmp_tab2;
    j = j+1;
end
end
```

B.2 Simulations

B.2.1 Dynamic causal modelling

```
function [dot_x] = dynamique(t,x,NROI,choix_reseau)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Crée une dynamique associée à un réseau pour le modèle DCM
% INPUTS: t = pas de temps considéré pour la dynamique
%         x = condition initiale du système dynamique
%         NROI = nombre de régions à considérer
%         choix_reseau = choix du réseau
% OUTPUT: dot_x = dynamique associée au réseau considére
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    switch choix_reseau
        case 1
            A = [-1.1450    0.1466    0.0075    0.0238    0.0000    0.1048    0.1446 ...
                   0.1913    0.0000    0.1758
                0.1780   -1.1450    0.1909    0.1436    0.0000    0.0000    0.1696 ...
                       0.0000    0.1865    0.0153
                0.1470    0.1313   -1.1450    0.0175    0.0575    0.0651    0.0000 ...
                       0.0009    0.1297    0.1010
                0.1021    0.1326    0.1169   -1.1450    0.0374    0.1238    0.0000 ...
                       0.0000    0.0061    0.0342
                0.0929    0.1052    0.0000    0.0200   -1.1450    0.0000    0.1646 ...
                       0.0000    0.0000    0.0182
                0.0058    0.0000    0.0000    0.0795    0.0557   -1.1450    0.0000 ...
                       0.0892    0.1143    0.0165
                0.0850    0.1551    0.1060    0.1301    0.0000    0.0149   -1.1450 ...
                       0.1748    0.1631    0.0509
                0.0524    0.0000    0.0000    0.1887    0.0079    0.1701    0.0000   ...
                    -1.1450    0.0632    0.1182
                0.0000    0.1926    0.1678    0.0500    0.1875    0.0000    0.1289 ...
                       0.0000   -1.1450    0.0557
                0.0000    0.0028    0.1156    0.0000    0.0236    0.0635    0.0000 ...
                       0.0228    0.0000   -1.1450];

            B = zeros(NROI,NROI);
            B(NROI-9,2) = 0.1427;
            B(3,NROI) = 0.1315;
            B(7,4) = 0.0000;

            C = ...
                0.01*[0.8199;0.9545;0.4784;0.7478;0.7004;0.3317;0.1237;0.9127;0.8420;0.9525];

        case 2
            A = [-0.6993  0.0000  0.0000  0.0000  0.0000  0.0876  0.0000  0.0000  ...
                0.0000  0.0000;
                0.0000 -0.8719  0.0000  0.0000  0.0000  0.0000  0.0000  0.1914  ...
                    0.4956  0.0000;
                0.0000  0.4432 -0.9111  0.0000  0.0000  0.0000  0.0298  0.0000  ...
```

```
                    0.2619  0.0000;
36                 0.0000  0.0000  0.0000 -0.6384  0.0854  0.0000  0.0000  0.2361  ...
                    0.0000  0.0074;
37                 0.0000  0.0000  0.0000  0.1394 -0.8006  0.0000  0.0000  0.0000  ...
                    0.3695  0.0783;
38                 0.0000  0.0000  0.0000  0.2912  0.0000 -0.5276  0.0000  0.0000  ...
                    0.0000  0.0000;
39                 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000 -0.7623  0.0000  ...
                    0.0000  0.0000;
40                 0.0000  0.0138  0.1000  0.0000  0.0000  0.0000  0.0000 -0.5768  ...
                    0.0000  0.0298;
41                 0.0000  0.0000  0.1714  0.0120  0.0000  0.0000  0.0000  0.0000 ...
                    -0.5201  0.0000;
42                 0.2839  0.0000  0.0000  0.0000  0.0000  0.0000  0.1900  0.0000  ...
                    0.0000 -0.5551];
43
44              B = zeros(NROI,NROI);
45              B(NROI-9,2) = 0.1427;
46              B(3,NROI) = 0.1315;
47              B(7,4) = 0.4709;
48
49              C = ...
                    0.01*[0.9055;1.0194;0.5029;0.8020;0.7894;0.4198;0.1924;0.9633;0.9043;0.9676];
50
51      end
52
53      u = 1+0.5*sin(5*t);
54      dot_x = A*x + u*B*x + C*u;
55  end
```

```
1   function [dydt] = systeme_hemodynamique(t,y,x_NS,t_NS)
2
3   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4   % Calcule les équations hémodynamiques du modèle DCM
5   % INPUTS: t = pas de temps considéré pour la dynamique
6   %         y = condition initiale des équations hémodynamiques
7   %         x_NS = dynamique retournée par l'intégration de la fonction
8   %                dynamique.m
9   %         t_NS = discrétisation du temps retournée par l'intégration de la
10  %                fonction dynamique.m
11  % OUTPUT: dot_x = équations d'états hémodynamiques
12  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13
14      % ___ Equations d'état neurovasculaire: ___ %
15      kappa = 0.65;
16      gamma = 0.41;
17
18      % Recherche du temps correspondant à t
19      i = 1;
20      while t>=t_NS(i)
21          i = i + 1;
22      end
```

```
23
24      % Interpolation linéaire
25      interp = x_NS(i)+(t-t_NS(i-1))*(x_NS(i)-x_NS(i-1))/(t_NS(i)-t_NS(i-1));
26
27      dydt(1,1) = interp-kappa*y(1)-gamma*(y(3)-1);
28      dydt(2,1) = y(1);
29
30      % ___ Modèle Ballon: ___ %
31      alpha = 0.32;
32      tau = 0.98;
33      E0 = 0.8;
34
35      dydt(3,1) = (y(2)-y(3).^(1/alpha))/tau;
36      dydt(4,1) = (y(2)*(1-(1-E0)^(1/y(2)))/E0-y(3)^(1/alpha)*y(4)/y(3))/tau;
37  end
```

B.2.2 Création des jeux de données

```
1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2   %                          Simulation des réseaux                           %
3   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4   close all
5   clear all
6   clc
7   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8   % Pour obtenir deux jeux de données différents, le code est à lancer deux
9   % fois en changeant la variable 'choix_reseau'
10  %
11  % Variables:
12  % choix_reseau = réseau considéré (1 ou 2)
13  choix_reseau = 2;
14  % write = booléen concernant le stockage des données
15  write = 0;
16  % nsub = nombre de patients considérés
17  nsub = 10;
18  % NROI = nombre de régions considérées
19  NROI = 10;
20  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21
22  for count = 1:nsub
23
24      % ___ Paramètres: ___ %
25      pas = 2.46;
26      intervalle = [0:pas:500];
27      intervalle_t = [0:pas:495];
28
29      x_0 = rand(NROI,1);
30      s_0 = rand(NROI,1);
31      f_0 = rand(NROI,1);
32      v_0 = rand(NROI,1);
33      q_0 = rand(NROI,1);
```

```

    V0 = 0.02;
    E0 = 0.8;

    k1 = 7*E0;
    k2 = 2;
    k3 = 2*E0-0.2;

    % ___ Calculs: ___ %

    % Calcul des équations d'état neuronal (Neural State equation)
    [t_NS,x_NS]=ode45(@(t_NS,x_NS) dynamique(t_NS,x_NS,NROI,choix_reseau)...
        ,intervalle,x_0,NROI);

    % Ajout de bruit à l'état x_NS
    x_NS = x_NS + 0.5*10^(-1)*x_NS.*randn(size(x_NS,1),size(x_NS,2));

    % Calcul du modèle hémodynamique (Hemodynamic model)
    %  et du signal BOLD
    for i=1:NROI
        y_0 = [s_0(i);f_0(i);v_0(i);q_0(i)];
        [t,y]=ode45(@(t,y) systeme_hemodynamique(t,y,x_NS(:,i),t_NS),...
            intervalle_t,y_0);

        BOLD(:,i) = V0*(k1*(1-y(:,4))+k2*(1-y(:,4)./y(:,3))+k3*(1-y(:,3)));
    end

    % Ecriture des simulations dans un fichier
    if write == 1
        csvwrite(sprintf('reseau%i_sub%i.csv',choix_reseau,count),BOLD)
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Simulations: Figures                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
nsub = 10;
pas = 2.46;
sub1_etat1 = load('reseau1_sub1.csv');
sub1_etat2 = load('reseau2_sub1.csv');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                             Signal BOLD                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
BOLD1 = sub1_etat1;
BOLD2 = sub1_etat2;
t = 1:pas:pas*size(BOLD1,1);

figure
for j=1:size(BOLD1,2)
    plot(t(21:end),BOLD1(21:end,j))
    hold on
end
title('Simulation du signal BOLD','interpreter','latex','fontsize',15)
xlabel('Fr\''equence d''\''echantillonage (toutes les $2.46$ ...
    secondes)','interpreter','latex','fontsize',15)
ylabel('BOLD','interpreter','latex','fontsize',15)

figure
for j=1:size(BOLD2,2)
    % Shift des données de 3.5*10^(-3) vers le bas
    plot(t(21:end),BOLD2(21:end,j)-3.5*10^(-3))
    hold on
end
title('Simulation du signal BOLD','interpreter','latex','fontsize',15)
xlabel('Fr\''equence d''\''echantillonage (toutes les $2.46$ ...
    secondes)','interpreter','latex','fontsize',15)
ylabel('BOLD','interpreter','latex','fontsize',15)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Spectre et Mode dominant                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[modes_etat1,lambda_etat1,scal1] = DMD(sub1_etat1',pas,1);
[modes_etat2,lambda_etat2,scal2] = DMD(sub1_etat2',pas,1);

valp_etat1 = (lambda_etat1);
valp_etat2 = (lambda_etat2);

%%%%%%%%%%%
% Spectre %
%%%%%%%%%%%
figure
for i=1:size(valp_etat1,2)
   plot(real(valp_etat1(i)),imag(valp_etat1(i)),'xb');
   hold on
   plot(real(valp_etat2(i)),imag(valp_etat2(i)),'xr');
end
axis equal
title('Spectre de la dynamique','interpreter','latex','fontsize',15)
xlabel('$Re(\lambda_{j})$','interpreter','latex','fontsize',15)
ylabel('$Im(\lambda_{j})$','interpreter','latex','fontsize',15)
legend('Réseau 1','Réseau 2','Location','southwest')

%%%%%%%%%%%%%%%%%
% Mode dominant %
%%%%%%%%%%%%%%%%%
```

```
71
72  modedom(:,1) = real(modes_etat1(:,end-1));
73  modedom(:,2) = real(modes_etat2(:,end-1));
74
75  figure
76  b = bar(modedom);
77  title('Composantes du mode dominant $g^d$','interpreter','latex','fontsize',15)
78  legend('Réseau 1','Réseau 2','Location','northwest')
79  xlabel('$g_i^d$','interpreter','latex','fontsize',15)
80  ylabel('$Re(g_{i}^d)$','interpreter','latex','fontsize',15)
81
82  for i=1:nsub
83
84      fichier1 = sprintf('reseau1_sub%d.csv',i);
85      fichier2 = sprintf('reseau2_sub%d.csv',i);
86      eval(sprintf('sub%d_etat1 = load(fichier1);',i))
87      eval(sprintf('sub%d_etat2 = load(fichier2)-3.5*10^(-3);',i))
88
89      [modes_etat1,lambda_etat1,scal1] = DMD(eval(sprintf('sub%i_etat1',i))',pas,1);
90      [modes_etat2,lambda_etat2,scal2] = DMD(eval(sprintf('sub%i_etat2',i))',pas,1);
91
92      mat_mdom(((i-1)*4)+1,:) = modes_etat1(:,end-1);
93      mat_mdom(((i-1)*4)+2,:) = modes_etat2(:,end-1);
94  end
95
96  ROI1 = 1;
97  ROI2 = 2;
98  ROI3 = 3;
99
100 figure
101 for k = 1:nsub
102     p1 = plot3(real(mat_mdom(((k-1)*4)+1,ROI1)),real(mat_mdom(((k-1)*4)...
103         +1,ROI2)),real(mat_mdom(((k-1)*4)+1,ROI3)),'*b');
104     hold on
105     p2 = plot3(real(mat_mdom(((k-1)*4)+2,ROI1)+0.05),real(mat_mdom(((k-1)*4)...
106         +2,ROI2)+0.05),real(mat_mdom(((k-1)*4)+2,ROI3)+0.05),'*r');
107     text(real(mat_mdom(((k-1)*4)+1,ROI1)+0.05),real(mat_mdom(((k-1)*4)...
108         +1,ROI2)+0.05),real(mat_mdom(((k-1)*4)+1,ROI3)+0.05),num2str(k),'Color','k');
109     text(real(mat_mdom(((k-1)*4)+2,ROI1)+0.09),real(mat_mdom(((k-1)*4)...
110         +2,ROI2)+0.09),real(mat_mdom(((k-1)*4)+2,ROI3)+0.09),num2str(k),'Color','k');
111 end
112
113 xlabel('ROI 1','interpreter','latex','fontsize',15)
114 ylabel('ROI 2','interpreter','latex','fontsize',15)
115 zlabel('ROI 3','interpreter','latex','fontsize',15)
116 title('Repr\''esentation de $Re(g^d)$ dans les trois premi\`eres ...
        ROI','interpreter','latex','fontsize',15)
117 grid on
118 legend([p1 p2],'Réseau 1','Réseau 2','Location','NorthWest')
```

B.2.3 Création du jeu de variables et classification

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Simulations: Création du jeu de variables pour la classification   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pour obtenir deux jeux de données différents, le code est à lancer deux
% fois en changeant la variable 'choix_reseau'
%
% Variables:
% nsub = nombre de patients considérés
nsub = 10;
% tri = tri lié à l'algorithme DMD: 1 = tri sur les valeurs propres
%                                   2 = tri sur les paramètres de scaling
tri = 1;
% vdata = sélection et stockage des données dans un fichier 'simul_vdata.csv'
prompt={'Valeur ''vdata'''};
name = 'Fichier données';
vdata = inputdlg(prompt,name,[1 30]);
vdata=str2num(vdata{1});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pas = 2.46;

% ___ Chargement des données: ___ %
for patient=1:nsub

    fichier1 = sprintf('reseau1_sub%d.csv',patient);
    fichier2 = sprintf('reseau2_sub%d.csv',patient);

    eval(sprintf('sub%d_etat1 = load(fichier1);',patient))
    eval(sprintf('sub%d_etat2 = load(fichier2)-3.5*10^(-3);',patient))
end

for i=1:nsub
    % ___ Calcul de DMD: ___ %
    [modes_etat1,lambda_etat1,scal1] = DMD(eval(sprintf('sub%i_etat1',i))',pas,tri);
    [modes_etat2,lambda_etat2,scal2] = DMD(eval(sprintf('sub%i_etat2',i))',pas,tri);

    for l=1:10
        for c=1:10
            smode1(l,c) = scal1(c)*modes_etat1(l,c);
            smode2(l,c) = scal2(c)*modes_etat2(l,c);
        end
    end

    mdom1 = modes_etat1(:,end-1);
    mdom2 = modes_etat2(:,end-1);

    moy_abs_valp1 = abs(mean(lambda_etat1));
    moy_abs_valp2 = abs(mean(lambda_etat2));

```

```
    tmdom1 = transpose(mdom1);
    tmdom2 = transpose(mdom2);

    switch vdata
        case 1
             vclass1 = [moy_abs_valp1 real(tmdom1) imag(tmdom1)];
             vclass2 = [moy_abs_valp2 real(tmdom2) imag(tmdom2)];
        case 2
             vclass1 = [real(tmdom1([2 3 4 5 6 8 9 10])) imag(tmdom1(2))];
             vclass2 = [real(tmdom2([2 3 4 5 6 8 9 10])) imag(tmdom2(2))];
        case 3
            vclass1 = [moy_abs_valp1 abs(tmdom1) angle(tmdom1)];
            vclass2 = [moy_abs_valp2 abs(tmdom2) angle(tmdom2)];
        case 4
            vclass1 = [abs(tmdom1([3,5,6,9])) angle(tmdom1([1,5,6,8]))];
            vclass2 = [abs(tmdom2([3,5,6,9])) angle(tmdom2([1,5,6,8]))];
        case 5
            vclass1 = [abs(tmdom1([3,5,6,9])) angle(tmdom1([1,5,6,8])) real(tmdom1([2 ...
                3 4 5 6 8 9 10])) imag(tmdom1(2))];
            vclass2 = [abs(tmdom2([3,5,6,9])) angle(tmdom2([1,5,6,8])) real(tmdom1([2 ...
                3 4 5 6 8 9 10])) imag(tmdom1(2))];
    end
    eval(sprintf('varbre%d = [vclass1;vclass2];',i))

end

% ___ Definition des classes: ___ %
varbre = varbre1;
for z = 2:nsub
    varbre = eval(sprintf('[varbre;varbre%d];',z));
end
class = [0; 1; 0; 1; 0; 1; 0; 1; 0; 1; 0; 1; 0; 1; 0; 1; 0; 1; 0; 1];
varbre = [varbre class];
nom_fichier_data = sprintf('simul_%i.csv',vdata);
csvwrite(nom_fichier_data,varbre)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Simulations: Classification Extra-Trees                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
% nsub = nombre de patients considérés
nsub = 10;
% DATA = fichier de données à importer (variables à considérer et classes)
DATA = single(load('simul_5.csv'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nb_class = size(AG,1);

```

```
% X = caractéristiques sur lesquelles la classification se base
% Y = variable cible (classes des patients)
X = DATA(:,1:end-1);
Y = DATA(:,end);

% ___ Paramètres de la méthode: ___ %

rtensparam = init_extra_trees()
% Nombre d'abres créés
rtensparam.nbterms = 100;
% Définition de K (=sqrt(nombre de caractéristiques disponibles))
rtensparam.rtparam.adjustdefaultk = 0;
rtensparam.rtparam.extratreesk = floor(sqrt(size(X,2)))+1;

% ___ Cross-validation: ___ %

predictions1 = zeros(nsub,nb_class-1);
predictions2 = zeros(nsub,nb_class-1);
vimportantes = zeros(size(X,2),nb_class);
for num_patient = 1:nsub
    % index_test = indices du patient utilisé pour tester l'arbre
    %              (le patient change toutes les 4 lignes)
    index_test = (num_patient-1)*2+1:num_patient*2;

    % ls = ensemble d'apprentissage
    ls = int32(1:nsub*2);
    ls(index_test) = [];
    ls = int32(ls);
    % w = poids des objets
    w = [];

    % XTS = ensemble test
    XTS = DATA(index_test,1:end-1);

    [YPRED1] = rtenslearn_c(X,Y,ls,w,rtensparam,XTS,1);
    predictions1(index_test,:) = YPRED1;

    [tree, var_imp] = rtenslearn_c(X,Y,ls,w,rtensparam,1);
    YPRED2 = rtenspred(tree,XTS);
    predictions2(index_test,:) = YPRED2;

    vimportantes = vimportantes+var_imp;
end
vimportantes = vimportantes/nsub;

% ___ Matrice de confusion: ___ %

figure
plotconfusion(Y',predictions1')
set(findobj(gca,'type','text'),'fontsize',14)
xt = get(gca, 'XTick');
set(gca, 'FontSize', 15)
title('Matrice de confusion','interpreter','latex','fontsize',15)
```

```
xlabel('Classes r\''eelles','interpreter','latex','fontsize',15)
ylabel('Classes pr\''edites','interpreter','latex','fontsize',15)

figure
bar(vimportantes)
title('Variables d''importance','interpreter','latex','fontsize',15)
xlabel('Variables de classification','interpreter','latex','fontsize',15)

[c,cm,ind,per] = confusion(Y',predictions1');

disp('Pourcentage matrice de confusion:')
per

% ___ ROC Curve: ___ %

figure
for k=1:nb_class-1
    posclass = 1;
    % labels = classes réelles
    % scores = classes prédites
    % posclass = positive class label
    [X,Y,T,AUC] = perfcurve(Y,predictions1,posclass);
    disp(['Area class ' num2str(k) ' = ' num2str(AUC)]);
    plot(X,Y)
    hold on
end

t=0:0.1:1;
plot(t,t,'--k')
xlabel('Taux de faux positifs','interpreter','latex','fontsize',15)
ylabel('Taux de vrais positifs','interpreter','latex','fontsize',15)
title('Courbe ROC','interpreter','latex','fontsize',15)
legend(['Courbe ROC  (AUC = ' num2str(roundn(AUC,-4)) ')'],'Location','southeast')
```

B.3 Données IRMf

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Données IRMf: Figures                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
% filtre = choix des fichiers de donnees
%  0 = reduced
%  1 = reduced_GS
%  2 = reduced_GS_filtered
filtre = 2;
% POI = patient utilisé pour les images
```

```
POI = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Chargement des fichiers                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nb_patients = 18;
boucle = 1:nb_patients;
% Il manque le patient 4 pour le premier jeu de données donc on le retire
% de la boucle
if filtre == 0
   boucle = [1:3 5:nb_patients];
   nb_patients = nb_patients-1;
end

for patient = boucle
    switch filtre
        case 0
            nom_fichier = sprintf('datafile/Mat_data_reduced_sub%d.mat',patient);
        case 1
            nom_fichier = sprintf('datafile/Mat_data_reduced_GS_sub%d.mat',patient);
        case 2
            nom_fichier = ...
                sprintf('datafile/Mat_data_reduced_GS_filtered_sub%d.mat',patient);
    end
    load(nom_fichier);
    matrice = Mat_data_reduced;

    eval(sprintf('sub%d_consc = matrice{1};',patient))
    eval(sprintf('sub%d_moyen = matrice{2};',patient))
    eval(sprintf('sub%d_profo = matrice{3};',patient))
    eval(sprintf('sub%d_revei = matrice{4};',patient))
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                Signal BOLD                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pas = 2.46;

% ___ Patient conscient ___ %
t = 1:pas:pas*size(eval(sprintf('sub%d_consc',POI)),1);

figure
for i=1:size(eval(sprintf('sub%d_consc',POI)),2)
    plot(t,eval(sprintf('sub%d_consc(:,i)',POI)))
    hold on
end
title('Signal BOLD d''un patient dans l''\''etat "conscient" ',...
    'interpreter','latex','fontsize',15)
xlabel('Fr\''equence d''\''echantillonage (toutes les $2.46$ secondes)',...
    'interpreter','latex','fontsize',15)
ylabel('BOLD','interpreter','latex','fontsize',15)
```

```
67
68  % ___ Patient moyennement endormi ___ %
69  t = 1:pas:pas*size(eval(sprintf('sub%d_moyen',POI)),1);
70
71  figure
72  for i=1:size(eval(sprintf('sub%d_moyen',POI)),2)
73      plot(t,eval(sprintf('sub%d_moyen(:,i)',POI)))
74      hold on
75  end
76  title('Signal BOLD d''un patient dans l''\''etat "moyennement endormi" ',...
77      'interpreter','latex','fontsize',15)
78  xlabel('Fr\''equence d''\''echantillonage (toutes les $2.46$ secondes)',...
79      'interpreter','latex','fontsize',15)
80  ylabel('BOLD','interpreter','latex','fontsize',15)
81
82  % ___ Patient profondément endormi ___ %
83  t = 1:pas:pas*size(eval(sprintf('sub%d_profo',POI)),1);
84
85  figure
86  for i=1:size(eval(sprintf('sub%d_profo',POI)),2)
87      plot(t,eval(sprintf('sub%d_profo(:,i)',POI)))
88      hold on
89  end
90  title('Signal BOLD d''un patient dans l''\''etat "profond\''ement endormi" ',...
91      'interpreter','latex','fontsize',15)
92  xlabel('Fr\''equence d''\''echantillonage (toutes les $2.46$ secondes)',...
93      'interpreter','latex','fontsize',15)
94  ylabel('BOLD','interpreter','latex','fontsize',15)
95
96  % ___ Patient en phase de réveil ___ %
97  t = 1:pas:pas*size(eval(sprintf('sub%d_revei',POI)),1);
98
99  figure
100 for i=1:size(eval(sprintf('sub%d_revei',POI)),2)
101     plot(t,eval(sprintf('sub%d_revei(:,i)',POI)))
102     hold on
103 end
104 title('Signal BOLD d''un patient dans l''\''etat "en phase de r\''eveil" ',...
105     'interpreter','latex','fontsize',15)
106 xlabel('Fr\''equence d''\''echantillonage (toutes les $2.46$ secondes)',...
107     'interpreter','latex','fontsize',15)
108 ylabel('BOLD','interpreter','latex','fontsize',15)
109
110 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
111 %                          Spectre et Mode dominant                             %
112 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
113
114 % ___ Définition des couleurs ___ %
115 col_consc = [0.9290, 0.6940, 0.1250];
116 col_moyen = [0.9294, 0.1373, 0.1373];
117 col_profo = [0.4660, 0.6740, 0.1880];
118 col_revei = [0, 0.4470, 0.7410];
119
```

```
[modes_consc,lambda_consc,scal1] = DMD(eval(sprintf('sub%i_consc',POI))',pas,1);
[modes_moyen,lambda_moyen,scal2] = DMD(eval(sprintf('sub%i_moyen',POI))',pas,1);
[modes_profo,lambda_profo,scal3] = DMD(eval(sprintf('sub%i_profo',POI))',pas,1);
[modes_revei,lambda_revei,scal4] = DMD(eval(sprintf('sub%i_revei',POI))',pas,1);

%%%%%%%%%%
% Spectre %
%%%%%%%%%%
figure
for i=1:size(lambda_consc,2)
   plot(real(lambda_consc(i)),imag(lambda_consc(i)),'x','color',col_consc);
   hold on
   plot(real(lambda_moyen(i)),imag(lambda_moyen(i)),'x','color',col_moyen);
   plot(real(lambda_profo(i)),imag(lambda_profo(i)),'x','color',col_profo);
   plot(real(lambda_revei(i)),imag(lambda_revei(i)),'x','color',col_revei);
end
axis equal
title('Spectre de la dynamique','interpreter','latex','fontsize',15)
xlabel('$Re(\lambda_{j})$','interpreter','latex','fontsize',15)
ylabel('$Im(\lambda_{j})$','interpreter','latex','fontsize',15)
legend('Conscient','Moyennement endormi','Profondément endormi','Phase de ...
    réveil','Location','southwest')

%%%%%%%%%%%%%%%%
% Mode dominant %
%%%%%%%%%%%%%%%%

modedom(:,1) = real(modes_consc(:,end));
modedom(:,2) = real(modes_moyen(:,end));
modedom(:,3) = real(modes_profo(:,end));
modedom(:,4) = real(modes_revei(:,end));

figure
b = bar(modedom,'FaceColor','flat');
set(b(1),'FaceColor',col_consc);
set(b(2),'FaceColor',col_moyen);
set(b(3),'FaceColor',col_profo);
set(b(4),'FaceColor',col_revei);
title('Composantes du mode dominant $g^d$','interpreter','latex','fontsize',15)
legend('Conscient','Moyennement endormi','Profondément endormi','Phase de ...
    réveil','Location','northwest')
xlabel('$g_i^d$','interpreter','latex','fontsize',15)
ylabel('$Re(g_{i}^d)$','interpreter','latex','fontsize',15)

mat_mdom = zeros(nb_patients*4,28);
for i = boucle
    [modes_consc,lambda_consc,scal1] = DMD(eval(sprintf('sub%i_consc',i))',pas,1);
    [modes_moyen,lambda_moyen,scal2] = DMD(eval(sprintf('sub%i_moyen',i))',pas,1);
    [modes_profo,lambda_profo,scal3] = DMD(eval(sprintf('sub%i_profo',i))',pas,1);
    [modes_revei,lambda_revei,scal4] = DMD(eval(sprintf('sub%i_revei',i))',pas,1);

    % Mode dominant: associé à lambda = 0
    mat_mdom(((i-1)*4)+1,:) = modes_consc(:,end);
```

```
    mat_mdom(((i-1)*4)+2,:) = modes_moyen(:,end);
    mat_mdom(((i-1)*4)+3,:) = modes_profo(:,end);
    mat_mdom(((i-1)*4)+4,:) = modes_revei(:,end);
end

ROI1 = 1;
ROI2 = 2;
ROI3 = 3;

figure
for k = boucle %1:nb_patients
    p1 = plot3(real(mat_mdom(((k-1)*4)+1,ROI1)),real(mat_mdom(((k-1)*4)...
        +1,ROI2)),real(mat_mdom(((k-1)*4)+1,ROI3)),'*','color',col_consc);
    hold on
    p2 = plot3(real(mat_mdom(((k-1)*4)+2,ROI1)),real(mat_mdom(((k-1)*4)...
        +2,ROI2)),real(mat_mdom(((k-1)*4)+2,ROI3)),'*','color',col_moyen);
    p3 = plot3(real(mat_mdom(((k-1)*4)+3,ROI1)),real(mat_mdom(((k-1)*4)...
        +3,ROI2)),real(mat_mdom(((k-1)*4)+3,ROI3)),'*','color',col_profo);
    p4 = plot3(real(mat_mdom(((k-1)*4)+4,ROI1)),real(mat_mdom(((k-1)*4)...
        +4,ROI2)),real(mat_mdom(((k-1)*4)+4,ROI3)),'*','color',col_revei);
    % Décommenter pour afficher le numéro de chaque patient
    %text(real(mat_mdom(((k-1)*4)+1,ROI1)),real(mat_mdom(((k-1)*4)...
    %    +1,ROI2)),real(mat_mdom(((k-1)*4)+1,ROI3)),num2str(k),'Color','k');
    %text(real(mat_mdom(((k-1)*4)+2,ROI1)),real(mat_mdom(((k-1)*4)...
    %    +2,ROI2)),real(mat_mdom(((k-1)*4)+2,ROI3)),num2str(k),'Color','k');
    %text(real(mat_mdom(((k-1)*4)+3,ROI1)),real(mat_mdom(((k-1)*4)...
    %    +3,ROI2)),real(mat_mdom(((k-1)*4)+3,ROI3)),num2str(k),'Color','k');
    %text(real(mat_mdom(((k-1)*4)+4,ROI1)),real(mat_mdom(((k-1)*4)...
    %    +4,ROI2)),real(mat_mdom(((k-1)*4)+4,ROI3)),num2str(k),'Color','k');
end
xlabel('ROI 1','interpreter','latex','fontsize',15)
ylabel('ROI 2','interpreter','latex','fontsize',15)
zlabel('ROI 3','interpreter','latex','fontsize',15)
title('Repr\''esentation de $Re(g^d)$ dans les trois premi\`eres ...
    ROI','interpreter','latex','fontsize',15)
grid on
legend([p1 p2 p3 p4],'Conscient','Moyennement endormi','Profondément endormi',...
    'Phase de réveil','Location','NorthWest')
```

B.3.1 Création du jeu de variables

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Données IRMf: Création du jeu de variables pour la classification    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
% filtre = choix des fichiers de donnees
%  0 = reduced
```

```
%  1 = reduced_GS
%  2 = reduced_GS_filtered
filtre = 1;
% nb_patients = nombre de patients considérés (1<=nb_patients<=18)
nb_patients = 18;
% tri = tri lié à l'algorithme DMD: 1 = tri sur les valeurs propres
%                                   2 = tri sur les parametres de scaling
tri = 1;
% vdata = stocke les variables pour les arbres sans le fichier
%         'data_yOx.csv' avec y = filtre
prompt={'Valeur ''vdata'''};
name = 'Fichier données';
vdata = inputdlg(prompt,name,[1 30]);
vdata=str2num(vdata{1});
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pas = 2.46;

boucle = 1:nb_patients;
% Il manque le patient 4 pour le premier jeu de données donc on le retire
% de la boucle
if filtre == 0
   boucle = [1:3 5:nb_patients];
   nb_patients = nb_patients-1;
end

for patient = boucle
    switch filtre
        case 0
            nom_fichier = sprintf('datafile/Mat_data_reduced_sub%d.mat',patient);
        case 1
            nom_fichier = sprintf('datafile/Mat_data_reduced_GS_sub%d.mat',patient);
        case 2
            nom_fichier = ...
                sprintf('datafile/Mat_data_reduced_GS_filtered_sub%d.mat',patient);
    end
    load(nom_fichier);
    matrice = Mat_data_reduced;

    eval(sprintf('sub%d_consc = matrice{1};',patient))
    eval(sprintf('sub%d_moyen = matrice{2};',patient))
    eval(sprintf('sub%d_profo = matrice{3};',patient))
    eval(sprintf('sub%d_revei = matrice{4};',patient))
end

for i = boucle
    [modes_consc,lambda_consc,scal1] = DMD(eval(sprintf('sub%i_consc',i))',pas,tri);
    [modes_moyen,lambda_moyen,scal2] = DMD(eval(sprintf('sub%i_moyen',i))',pas,tri);
    [modes_profo,lambda_profo,scal3] = DMD(eval(sprintf('sub%i_profo',i))',pas,tri);
    [modes_revei,lambda_revei,scal4] = DMD(eval(sprintf('sub%i_revei',i))',pas,tri);

    for l=1:28
        for c=1:28
```

```
            smodec(l,c) = scal1(c)*modes_consc(l,c);
            smodem(l,c) = scal2(c)*modes_moyen(l,c);
            smodep(l,c) = scal3(c)*modes_profo(l,c);
            smoder(l,c) = scal4(c)*modes_revei(l,c);
        end
    end

    mdomc = modes_consc(:,end-1);
    mdomm = modes_moyen(:,end-1);
    mdomp = modes_profo(:,end-1);
    mdomr = modes_revei(:,end-1);

    abs_moy_valpc = abs(mean((lambda_consc)));
    abs_moy_valpm = abs(mean((lambda_moyen)));
    abs_moy_valpp = abs(mean((lambda_profo)));
    abs_moy_valpr = abs(mean((lambda_revei)));

    valpc = lambda_consc;
    valpm = lambda_moyen;
    valpp = lambda_profo;
    valpr = lambda_revei;

    tmdomc = transpose(mdomc);
    tmdomm = transpose(mdomm);
    tmdomp = transpose(mdomp);
    tmdomr = transpose(mdomr);

    tsmodec1 = transpose(smodec(:,end-2));
    tsmodem1 = transpose(smodem(:,end-2));
    tsmodep1 = transpose(smodep(:,end-2));
    tsmoder1 = transpose(smoder(:,end-2));

    norm_tsmodec2 = norm(transpose(smodec(:,end-3)));
    norm_tsmodem2 = norm(transpose(smodem(:,end-3)));
    norm_tsmodep2 = norm(transpose(smodep(:,end-3)));
    norm_tsmoder2 = norm(transpose(smoder(:,end-3)));

    norm_tsmodec3 = norm(transpose(smodec(:,end-4)));
    norm_tsmodem3 = norm(transpose(smodem(:,end-4)));
    norm_tsmodep3 = norm(transpose(smodep(:,end-4)));
    norm_tsmoder3 = norm(transpose(smoder(:,end-4)));

    norm_tsmodec = norm(transpose(smodec));
    norm_tsmodem = norm(transpose(smodem));
    norm_tsmodep = norm(transpose(smodep));
    norm_tsmoder = norm(transpose(smoder));

    tmp21 = norm_tsmodec^2-norm(tsmodec1)^2-norm_tsmodec2^2-norm_tsmodec3^2;
    tmp22 = norm_tsmodem^2-norm(tsmodem1)^2-norm_tsmodem2^2-norm_tsmodem3^2;
    tmp23 = norm_tsmodep^2-norm(tsmodep1)^2-norm_tsmodep2^2-norm_tsmodep3^2;
    tmp24 = norm_tsmoder^2-norm(tsmoder1)^2-norm_tsmoder2^2-norm_tsmoder3^2;

    switch vdata
```

```
116         case 1
117             if filter == 0
118                 vclass1 = [abs(tmdomc([1:3,5:7,9:28])) ...
                        angle(tmdomc([1,3:4,7:10,12,14:28]))];
119                 vclass2 = [abs(tmdomm([1:3,5:7,9:28])) ...
                        angle(tmdomm([1,3:4,7:10,12,14:28]))];
120                 vclass3 = [abs(tmdomp([1:3,5:7,9:28])) ...
                        angle(tmdomp([1,3:4,7:10,12,14:28]))];
121                 vclass4 = [abs(tmdomr([1:3,5:7,9:28])) ...
                        angle(tmdomr([1,3:4,7:10,12,14:28]))];
122             end
123             if filter == 1
124                 vclass1 = [abs(tmdomc([1:11,13:28])) ...
                        angle(tmdomc([1,2,4:13,15:19,23,24,26:28]))];
125                 vclass2 = [abs(tmdomm([1:11,13:28])) ...
                        angle(tmdomm([1,2,4:13,15:19,23,24,26:28]))];
126                 vclass3 = [abs(tmdomp([1:11,13:28])) ...
                        angle(tmdomp([1,2,4:13,15:19,23,24,26:28]))];
127                 vclass4 = [abs(tmdomr([1:11,13:28])) ...
                        angle(tmdomr([1,2,4:13,15:19,23,24,26:28]))];
128             end
129             if filter == 2
130                 vclass1 = [abs(tmdomc([1,3:24,27,28])) ...
                        angle(tmdomc([1:7,9:13,15:24,27,28]))];
131                 vclass2 = [abs(tmdomm([1,3:24,27,28])) ...
                        angle(tmdomm([1:7,9:13,15:24,27,28]))];
132                 vclass3 = [abs(tmdomp([1,3:24,27,28])) ...
                        angle(tmdomp([1:7,9:13,15:24,27,28]))];
133                 vclass4 = [abs(tmdomr([1,3:24,27,28])) ...
                        angle(tmdomr([1:7,9:13,15:24,27,28]))];
134             end
135
136         case 2
137             vclass1 = [real(mean(exp(valpc))) abs(tmdomc([1,10,14,22,26])) ...
                    angle(tmdomc([1,5,15,20])) norm_tsmodec mean(real((exp(valpc))).^2) ...
                    tmp21];
138             vclass2 = [real(mean(exp(valpm))) abs(tmdomm([1,10,14,22,26])) ...
                    angle(tmdomm([1,5,15,20])) norm_tsmodem mean(real((exp(valpm))).^2) ...
                    tmp22];
139             vclass3 = [real(mean(exp(valpp))) abs(tmdomp([1,10,14,22,26])) ...
                    angle(tmdomp([1,5,15,20])) norm_tsmodep mean(real((exp(valpp))).^2) ...
                    tmp23];
140             vclass4 = [real(mean(exp(valpr))) abs(tmdomr([1,10,14,22,26])) ...
                    angle(tmdomr([1,5,15,20])) norm_tsmoder mean(real((exp(valpr))).^2) ...
                    tmp24];
141
142         case 3
143             vclass1 = [real(mean(exp(valpc))) abs(tmdomc([1,10,14,22,26])) ...
                    angle(tmdomc([1,5,15,20])) norm_tsmodec mean(real(exp(valpc).^2)) ...
                    mean(real(exp(valpc).^3)) tmp21];
144             vclass2 = [real(mean(exp(valpm))) abs(tmdomm([1,10,14,22,26])) ...
                    angle(tmdomm([1,5,15,20])) norm_tsmodem mean(real(exp(valpm).^2)) ...
                    mean(real(exp(valpm).^3)) tmp22];
```

```
145             vclass3 = [real(mean(exp(valpp))) abs(tmdomp([1,10,14,22,26])) ...
                    angle(tmdomp([1,5,15,20])) norm_tsmodep mean(real(exp(valpp).^2)) ...
                    mean(real(exp(valpp).^3)) tmp23];
146             vclass4 = [real(mean(exp(valpr))) abs(tmdomr([1,10,14,22,26])) ...
                    angle(tmdomr([1,5,15,20])) norm_tsmoder mean(real(exp(valpr).^2)) ...
                    mean(real(exp(valpr).^3)) tmp24];
147
148         case 4
149             vclass1 = [abs_moy_valpc mean(abs(valpc)) abs(tmdomc([1,10,14,22,26])) ...
                    angle(tmdomc([1,5,15,20])) norm_tsmodec mean(abs(valpc).^2) tmp21 ...
                    imag(tmdomc(25)) mean(imag(valpc).^2) mean(real(valpc).^2)];
150             vclass2 = [abs_moy_valpm mean(abs(valpm)) abs(tmdomm([1,10,14,22,26])) ...
                    angle(tmdomm([1,5,15,20])) norm_tsmodem mean(abs(valpm).^2) tmp22 ...
                    imag(tmdomm(25)) mean(imag(valpm).^2) mean(real(valpm).^2)];
151             vclass3 = [abs_moy_valpp mean(abs(valpp)) abs(tmdomp([1,10,14,22,26])) ...
                    angle(tmdomp([1,5,15,20])) norm_tsmodep mean(abs(valpp).^2) tmp23 ...
                    imag(tmdomp(25)) mean(imag(valpp).^2) mean(real(valpp).^2)];
152             vclass4 = [abs_moy_valpr mean(abs(valpr)) abs(tmdomr([1,10,14,22,26])) ...
                    angle(tmdomr([1,5,15,20])) norm_tsmoder mean(abs(valpr).^2) tmp24 ...
                    imag(tmdomr(25)) mean(imag(valpr).^2) mean(real(valpr).^2)];
153
154         case 5
155             vclass1 = [mean(real(exp(valpc))) ...
                    abs(tmdomc([3,5,6,7,10,15,17,18,21,22,25]))];
156             vclass2 = [mean(real(exp(valpm))) ...
                    abs(tmdomm([3,5,6,7,10,15,17,18,21,22,25]))];
157             vclass3 = [mean(real(exp(valpp))) ...
                    abs(tmdomp([3,5,6,7,10,15,17,18,21,22,25]))];
158             vclass4 = [mean(real(exp(valpr))) ...
                    abs(tmdomr([3,5,6,7,10,15,17,18,21,22,25]))];
159
160     end
161     eval(sprintf('varbre%d = [vclass1;vclass2;vclass3;vclass4];',i))
162 end
163
164 % Création de la matrice contenant les données pour les arbres
165 varbre = varbre1;
166 for z = boucle(2:end) %2:nb_patients
167     varbre = eval(sprintf('[varbre;varbre%d];',z));
168 end
169
170 % Création de la matrice contenant les classes associées aux données
171 %       Classe 1: 1 0 0 0
172 %       Classe 2: 0 1 0 0
173 %       Classe 3: 0 0 1 0
174 %       Classe 4: 0 0 0 1
175 class = [1 0 0 0; 0 1 0 0;0 0 1 0; 0 0 0 1];
176 for z = boucle(2:end)
177     class = [class;[1 0 0 0; 0 1 0 0;0 0 1 0; 0 0 0 1];];
178 end
179 varbre = [varbre class];
180 nom_fichier_data = sprintf('data_%i.csv',vdata);
181 csvwrite(nom_fichier_data,varbre)
```

```
fprintf('Data set: %i\n',vdata)
disp('Computation done')
```

B.3.2 Classifications

Extra-Trees

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Données IRMf: Classification Extra-Trees                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
% nsub = nombre de patients considérés
nsub = 18;
% AG = matrice de regroupement des classes, où le cas 4 classes:
%           AG = [1;2;3;4].
%      Exemple: regrouper classe 2 et 4: AG = [1 0; 2 4; 3 0]
AG = [1;2;3;4];
% DATA = fichier de données à importer (variables à considérer et classes)
DATA = single(load('data_1.csv'));
% label = libellé des classes considérées pour la ROC Curve.
%               Si legendInfo est vide, le libelé sera 'classe k'
label = {'Conscient','Moyennement endormi','Profondément endormi','Phase de réveil'};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Définition des couleurs
col_consc = [0.9290, 0.6940, 0.1250];
col_moyen = [0.9294, 0.1373, 0.1373];
col_profo = [0.4660, 0.6740, 0.1880];
col_revei = [0, 0.4470, 0.7410];
colors = [col_consc; col_moyen; col_profo; col_revei];

nb_class = size(AG,1);
tmp = find(AG,nb_class);
class = zeros(nb_class,'single');
for i=1:nb_class
    class(i,tmp(i)) = single(1);
end

% X = caractéristiques sur lesquelles la classification se base
% Y = variable cible (classes des patients)
X = DATA(:,1:end-4);
for i = 1:size(DATA,1)/4
    for j = 1:size(AG,1)
        for k = 1:size(AG,2)
            if(AG(j,k)~=0)
                Y(AG(j,k)+(i-1)*4,:) = class(j,:);
            end
```

```
        end
    end
end

% ___ Paramètres de la méthode: ___ %

rtensparam = init_extra_trees()
% Nombre d'arbres créés
rtensparam.nbterms = 100;
% Définition de K (=sqrt(nombre de caractéristiques disponibles))
rtensparam.rtparam.adjustdefaultk = 0;
rtensparam.rtparam.extratreesk = floor(sqrt(size(X,2)))+1;

% ___ Cross-validation: ___ %

predictions1 = zeros(4*nsub,nb_class);
predictions2 = zeros(4*nsub,nb_class);
vimportantes = zeros(size(X,2),nb_class);
for num_patient = 1:nsub
    % index_test = indices pour le patient utilisé pour tester l'arbre
    %              (le patient change toutes les 4 lignes)
    index_test = (num_patient-1)*4+1:num_patient*4;

    % ls = ensemble d'apprentissage
    ls = int32(1:nsub*4);
    ls(index_test) = [];
    ls = int32(ls);
    % w = poids des objets
    w = [];

    % XTS = ensemble test
    XTS = DATA(index_test,1:end-4);

    [YPRED1] = rtenslearn_c(X,Y,ls,w,rtensparam,XTS,1);
    predictions1(index_test,:) = YPRED1;

    [tree, var_imp] = rtenslearn_c(X,Y,ls,w,rtensparam,1);
    YPRED2 = rtenspred(tree,XTS);
    predictions2(index_test,:) = YPRED2;

    vimportantes = vimportantes+var_imp;
end
vimportantes = vimportantes/nsub;

% ___ Matrice de confusion ___ %

figure
plotconfusion(Y',predictions1')
set(findobj(gca,'type','text'),'fontsize',14)
xt = get(gca, 'XTick');
set(gca, 'FontSize', 15)
title('Matrice de confusion','interpreter','latex','fontsize',15)
xlabel('Classes r\''eelles','interpreter','latex','fontsize',15)
```

```
ylabel('Classes pr\''edites','interpreter','latex','fontsize',15)

figure
b=bar(vimportantes);
for i=1:size(label,2)
    set(b(i),'FaceColor',colors(i,:));
end
title('Variables d''importance','interpreter','latex','fontsize',15)
xlabel('Variables de classification','interpreter','latex','fontsize',15)
legend(label,'Location','northwest')

[c,cm,ind,per] = confusion(Y',predictions1');

disp('Pourcentage matrice de confusion:')
per

% ___ ROC Curve: ___ %

figure
for k=1:nb_class
    for i=1:nsub
        if k==1
            index = [(i-1)*4+2 (i-1)*4+3 (i-1)*4+4];
        else if k==2
                index = [(i-1)*4+1 (i-1)*4+3 (i-1)*4+4];
            else if k==3
                    index = [(i-1)*4+1 (i-1)*4+2 (i-1)*4+4];
                else
                    index = [(i-1)*4+1 (i-1)*4+2 (i-1)*4+3];
                end
            end
        end
        scores((i-1)*4+k,1) = predictions1((i-1)*4+k,k);
        scores(index,1) = predictions1(index,k);

        labels((i-1)*4+k,1) = 1;
        labels(index,1) = 0;
    end
    posclass = 1;
    % labels = classes réelles
    % scores = classes prédites
    % posclass = positive class label
    [X,Y,T,AUC] = perfcurve(labels,scores,posclass);
    disp(['Area class ' num2str(k) ' = ' num2str(AUC)]);
    plot(X,Y,'color',colors(k,:))
    hold on
    if isempty(label)
        legendInfo{k} = ['Class ' num2str(k) '  (AUC = ' num2str(roundn(AUC,-4)) ')'];
    else
        legendInfo{k} = [label{k} '  (AUC = ' num2str(roundn(AUC,-4)) ')'];
    end
end

```

```
t=0:0.1:1;
plot(t,t,'--k')
xlabel('Taux de faux positifs','interpreter','latex','fontsize',15)
ylabel('Taux de vrais positifs','interpreter','latex','fontsize',15)
title('Courbe ROC','interpreter','latex','fontsize',15)
legend(legendInfo,'Location','southeast')
```

Arbre de décision

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Données IRMf: Decision Tree                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
% nsub = nombre de patients considérés
nsub = 18;
% AG = matrice de regroupement des classes, où le cas 4 classes:
%           AG = [1;2;3;4].
%      Exemple: regrouper classe 2 et 4: AG = [1 0; 2 4; 3 0]
AG = [1;2;3;4];
% DATA = fichier de données à importer (variables à considérer et classes)
DATA = double(load('data_5.csv'));
% label = libellé des classes considérées pour la ROC Curve.
%               Si legendInfo est vide, le libelé sera 'classe k'
label = {'Eveillé','Moyennement endormi','Profondément endormi','Phase de réveil'};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Définition des couleurs
col_consc = [0.9290, 0.6940, 0.1250];
col_moyen = [0.9294, 0.1373, 0.1373];
col_profo = [0.4660, 0.6740, 0.1880];
col_revei = [0, 0.4470, 0.7410];
colors = [col_consc; col_moyen; col_profo; col_revei];

nb_class = size(AG,1);
tmp = find(AG,nb_class);
class = zeros(nb_class,'double');
for i=1:nb_class
    class(i,tmp(i)) = double(1);
end

% X = caractéristiques sur lesquelles la classification se base
% Y = variable cible (classes des patients)
X = DATA(:,1:end-4);

for i = 1:size(DATA,1)/4
    for j = 1:size(AG,1)
        for k = 1:size(AG,2)
```

```
            if(AG(j,k)~=0)
                Y(AG(j,k)+(i-1)*4,:) = class(j,:);
            end
        end
    end
end
class1 = [1 0 0 0];
class2 = [0 1 0 0];
class3 = [0 0 1 0];
class4 = [0 0 0 1];
for i=1:size(Y,1)
    if isequal(Y(i,:),class1)
        labels(i,1) = 0;
    elseif isequal(Y(i,:),class2)
        labels(i,1) = 1;
    elseif isequal(Y(i,:),class3)
        labels(i,1) = 2;
    else
        labels(i,1) = 3;
    end
end

% ___ Cross-validation: ___ %

predictions = zeros(4*nsub,nb_class);
for num_patient = 1:nsub
    % index_test = indices pour le patient utilisé pour tester l'arbre
    %               (le patient change toutes les 4 lignes)
    index_test = (num_patient-1)*4+1:num_patient*4;

    % ls = ensemble d'apprentissage
    ls = int32(1:nsub*4);
    ls(index_test) = [];
    ls = int32(ls);
    % w = poids des objets
    w = [];

    % XTS = ensemble test
    XTS = DATA(index_test,1:end-4);

    Xtrain = X;
    Ytrain = labels;
    Xtrain(index_test,:) = [];
    Ytrain(index_test,:) = [];

    tree = fitctree(Xtrain,Ytrain)
    [YPRED,score] = predict(tree,XTS);

    predictions(index_test,:) = score;
end

% ___ Matrice de confusion: ___ %

```

```
figure
plotconfusion(Y',predictions')
set(findobj(gca,'type','text'),'fontsize',14)
xt = get(gca, 'XTick');
set(gca, 'FontSize', 15)
title('Matrice de confusion','interpreter','latex','fontsize',15)
xlabel('Classes r\''eelles','interpreter','latex','fontsize',15)
ylabel('Classes pr\''edites','interpreter','latex','fontsize',15)

[c,cm,ind,per] = confusion(Y',predictions');

disp('Pourcentage matrice de confusion:')
per

% ___ ROC Curve: ___ %

figure
for k=1:nb_class
    for i=1:nsub
        if k==1
            index = [(i-1)*4+2 (i-1)*4+3 (i-1)*4+4];
        else if k==2
                index = [(i-1)*4+1 (i-1)*4+3 (i-1)*4+4];
            else if k==3
                    index = [(i-1)*4+1 (i-1)*4+2 (i-1)*4+4];
                else
                    index = [(i-1)*4+1 (i-1)*4+2 (i-1)*4+3];
                end
            end
        end
        scores((i-1)*4+k,1) = predictions((i-1)*4+k,k);
        scores(index,1) = predictions(index,k);

        labels((i-1)*4+k,1) = 1;
        labels(index,1) = 0;
    end
    posclass = 1;
    % labels = classes réelles
    % scores = classes prédites
    % posclass = positive class label
    [X,Y,T,AUC] = perfcurve(labels,scores,posclass);
    disp(['Area class ' num2str(k) ' = ' num2str(AUC)]);
    plot(X,Y,'color',colors(k,:))
    hold on
    if isempty(label)
        legendInfo{k} = ['Class ' num2str(k) '  (AUC = ' num2str(roundn(AUC,-4)) ')'];
    else
        legendInfo{k} = [label{k} '  (AUC = ' num2str(roundn(AUC,-4)) ')'];
    end
end

t=0:0.1:1;
plot(t,t,'--k')
```

```
xlabel('Taux de faux positifs','interpreter','latex','fontsize',15)
ylabel('Taux de vrais positifs','interpreter','latex','fontsize',15)
title('Courbe ROC','interpreter','latex','fontsize',15)
legend(legendInfo,'Location','southeast')
```

Régression logistique

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Données IRMf: Logistic regression                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
% nsub = nombre de patients considérés
nsub = 18;
% AG = matrice de regroupement des classes, où le cas 4 classes:
%           AG = [1;2;3;4].
%      Exemple: regrouper classe 2 et 4: AG = [1 0; 2 4; 3 0]
AG = [1;2;3;4];
% DATA = fichier de données à importer (variables à considérer et classes)
DATA = double(load('data_5.csv'));
% label = libellé des classes considérées pour la ROC Curve.
%               Si legendInfo est vide, le libelé sera 'classe k'
label = {'Eveillé','Moyennement endormi','Profondément endormi','Phase de réveil'};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Définition des couleurs
col_consc = [0.9290, 0.6940, 0.1250];
col_moyen = [0.9294, 0.1373, 0.1373];
col_profo = [0.4660, 0.6740, 0.1880];
col_revei = [0, 0.4470, 0.7410];
colors = [col_consc; col_moyen; col_profo; col_revei];

nb_class = size(AG,1);
tmp = find(AG,nb_class);
class = zeros(nb_class,'double');
for i=1:nb_class
    class(i,tmp(i)) = double(1);
end

% X = caractéristiques sur lesquelles la classification se base
% Y = variable cible (classes des patients)
X = DATA(:,1:end-4);
for i = 1:size(DATA,1)/4
    for j = 1:size(AG,1)
        for k = 1:size(AG,2)
            if(AG(j,k)~=0)
                Y(AG(j,k)+(i-1)*4,:) = class(j,:);
            end
```

```
        end
    end
end

% ___ Cross-validation: ___ %

predictions = zeros(4*nsub,nb_class);
for num_patient = 1:nsub
    num_patient
    % index_test = indices pour le patient utilisé pour tester la
    % régression (le patient change toutes les 4 lignes)
    index_test = (num_patient-1)*4+1:num_patient*4;

    % ls = ensemble d'apprentissage
    ls = int32(1:nsub*4);
    ls(index_test) = [];
    ls = int32(ls);
    % w = poids des objets
    w = [];

    % XTS = ensemble test
    XTS = DATA(index_test,1:end-4);

    [B,dev,stats] = mnrfit(X(ls,:),Y(ls,:));
    YPRED = mnrval(B,XTS);
    predictions(index_test,:) = YPRED;

end
% ___ Matrice de confusion: ___ %

figure
plotconfusion(Y',predictions')
set(findobj(gca,'type','text'),'fontsize',14)
xt = get(gca, 'XTick');
set(gca, 'FontSize', 15)
title('Matrice de confusion','interpreter','latex','fontsize',15)
xlabel('Classes r\''eelles','interpreter','latex','fontsize',15)
ylabel('Classes pr\''edites','interpreter','latex','fontsize',15)

[c,cm,ind,per] = confusion(Y',predictions');

disp('Pourcentage matrice de confusion:')
per

% ___ ROC Curve: ___ %

figure
for k=1:nb_class %4
    for i=1:nsub
        if k==1
            index = [(i-1)*4+2 (i-1)*4+3 (i-1)*4+4];
        else if k==2
                index = [(i-1)*4+1 (i-1)*4+3 (i-1)*4+4];
```

```
98              else if k==3
99                      index = [(i-1)*4+1 (i-1)*4+2 (i-1)*4+4];
100                 else
101                     index = [(i-1)*4+1 (i-1)*4+2 (i-1)*4+3];
102                 end
103             end
104         end
105         scores((i-1)*4+k,1) = predictions((i-1)*4+k,k);
106         scores(index,1) = predictions(index,k);
107 
108         labels((i-1)*4+k,1) = 1;
109         labels(index,1) = 0;
110     end
111     posclass = 1;
112     % labels = classes réelles
113     % scores = classes prédites
114     % posclass = positive class label
115     [X,Y,T,AUC] = perfcurve(labels,scores,posclass);
116     disp(['Area class ' num2str(k) ' = ' num2str(AUC)]);
117     plot(X,Y,'color',colors(k,:))
118     hold on
119     if isempty(label)
120         legendInfo{k} = ['Class ' num2str(k) '  (AUC = ' num2str(roundn(AUC,-4)) ')'];
121     else
122         legendInfo{k} = [label{k} '  (AUC = ' num2str(roundn(AUC,-4)) ')'];
123     end
124 end
125 
126 t=0:0.1:1;
127 plot(t,t,'--k')
128 xlabel('Taux de faux positifs','interpreter','latex','fontsize',15)
129 ylabel('Taux de vrais positifs','interpreter','latex','fontsize',15)
130 title('Courbe ROC','interpreter','latex','fontsize',15)
131 legend(legendInfo,'Location','southeast')
```